Matematyka stosowana

Matematyka obliczeniowa II

Piotr Krzyżanowski
http://www.mimuw.edu.pl/~przykry

Leszek Plaskota
http://www.mimuw.edu.pl/~leszekp

Uniwersytet Warszawski, 2013

Streszczenie. W skrypcie omawiamy - w zakresie szerszym niż w semestralnym wykładzie - zaawansowane metody rozwiązywania podstawowych zagadnień obliczeniowych spotykanych w praktycznej matematyce stosowanej: wielkich układów równań liniowych i nieliniowych, zagadnienia własnego i jemu pokrewnych oraz obliczaniu całek wielowymiarowych. Są to kanoniczne zagadnienia matematyki obliczeniowej, których wspólnym mianownikiem we współczesnym świecie jest to, że dotyczą obiektów bardzo dużego rozmiaru. Kurs wymaga podstawowej wiedzy z Analizy i z Algebry Liniowej i będzie przydatny wszystkim, którzy liczą się z tym, że będą musieli w przyszłości policzyć coś bardziej skomplikowanego niż współczynniki wielomianu interpolacyjnego...

Wersja internetowa wykładu:
http://mst.mimuw.edu.pl/lecture.php?lecture=mo2

(może zawierać dodatkowe materiały, ale nie jest już aktualizowana)

BY-NC-ND

Niniejsze materiały są dostępne na licencji Creative Commons 3.0 Polska:
Uznanie autorstwa — Użycie niekomercyjne — Bez utworów zależnych.

Copyright © P.Krzyżanowski, L.Plaskota, Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki, 2013. Niniejszy plik PDF został utworzony 25 listopada 2013.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską w ramach http://www.efs.gov.pl.

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Skład w systemie LaTeX, z wykorzystaniem m.in. pakietów beamer oraz listings. Szablony podręcznika i prezentacji: Piotr Krzyżanowski; koncept: Robert Dąbrowski.

Spis treści

Wprowadzenie . . . 6

1. Zagadnienie własne I . . . 8

1.1. Podstawy teoretyczne . . . 8
1.2. Teoria zaburzeń dla macierzy niesymetrycznych . . . 9
1.2.1. Wstępny przykład . . . 9
1.2.2. Wrażliwość wartości własnych . . . 10
1.2.3. Wrażliwość wektorów własnych . . . 12
1.3. Teoria zaburzeń dla macierzy symetrycznych . . . 13
1.3.1. Wrażliwość wartości własnych . . . 13
1.3.2. Wrażliwość wektorów własnych . . . 14

2. Zagadnienie własne II . . . 17

2.1. Uwagi wstępne . . . 17
2.1.1. Sprowadzanie macierzy do postaci Hessenberga . . . 18
2.2. Metoda Hymana . . . 19
2.3. Metoda potęgowa . . . 20
2.3.1. Definicja metody . . . 20
2.3.2. Analiza zbieżności . . . 20
2.4. Odwrotna metoda potęgowa i deflacja . . . 23

3. Zagadnienie własne III . . . 25

3.1. Iteracje podprzestrzeni . . . 25
3.1.1. Algorytm ogólny . . . 25
3.1.2. Iteracje ortogonalne . . . 26
3.2. Metoda QR . . . 27
3.2.1. Wyprowadzenie metody . . . 27
3.2.2. QR a iteracje ortogonalne . . . 28
3.2.3. Analiza zbieżności . . . 29
3.3. QR z przesunięciami i deflacja . . . 31

4. Zagadnienie własne IV . . . 33

4.1. Obroty Givensa . . . 33
4.1.1. Definicja obrotu . . . 33
4.1.2. Algorytm Gentlemana . . . 34
4.2. Metoda Jacobiego . . . 36
4.3. QR dla macierzy trójdiagonalnej . . . 38
4.4. Rozkład według wartości szczególnych (SVD) . . . 39
4.4.1. Twierdzenie o rozkładzie . . . 39
4.4.2. Dlaczego nie pomnożyć $A^T A$? . . . 40
4.4.3. SVD dla macierzy dwudiagonalnych . . . 41

5. Proste metody iteracyjne rozwiązywania układów równań liniowych . . . 44

5.1. Macierze rozrzedzone . . . 44
5.1.1. Przykłady macierzy rozrzedzonych . . . 45
5.2. Macierze rozrzedzone: implementacja . . . 50
5.3. Metody stacjonarne rozwiązywania układów równań liniowych . . . 53
5.3.1. Metoda Jacobiego . . . 55
5.3.2. Metoda Gaussa–Seidela . . . 60
5.3.3. Metoda SOR . . . 61

5.4. Metody blokowe . . . 66
5.5. Metody projekcji . . . 67
5.5.1. Metoda najszybszego spadku . . . 68
5.5.2. Metoda najmniejszego residuum . . . 70
6. Metody iteracyjne Kryłowa . . . 72
6.1. Metoda gradientów sprzężonych (CG) . . . 75
6.1.1. Implementacja . . . 76
6.2. Metoda GMRES . . . 81
6.2.1. Implementacja . . . 82
6.2.2. Wersja z restartem: GMRES(m) . . . 84
7. Ściskanie macierzy (preconditioning) . . . 85
7.1. Proste operatory ściskające . . . 86
7.2. Macierze spektralnie równoważne . . . 88
7.3. Metoda PCG . . . 90
7.4. Ściskanie dla GMRES . . . 92
7.5. Kryteria stopu metod iteracyjnych . . . 92
7.5.1. Kryterium błędu . . . 92
7.5.2. Kryterium residualne . . . 93
7.5.3. Kryterium postępu . . . 94
8. Przegląd innych metod rozwiązywania wielkich układów równań liniowych . . . 95
8.1. Inne metody Kryłowa . . . 95
8.1.1. Uogólnienia metody CG . . . 95
8.1.2. Metody nie oparte na minimalizacji . . . 98
8.2. Metody strukturalne . . . 98
8.3. Metody wielosiatkowe . . . 100
9. Układy równań nieliniowych. Metoda Newtona . . . 103
9.1. Szybkość zbieżności . . . 103
9.2. Metoda Banacha . . . 105
9.3. Metoda Newtona . . . 107
9.3.1. Analiza zbieżności metody Newtona . . . 107
9.3.2. Implementacja metody Newtona . . . 109
10. Wariacje na temat metody Newtona . . . 112
10.1. Twierdzenie o istnieniu rozwiązań . . . 112
10.2. Obniżanie kosztu iteracji . . . 112
10.2.1. Uproszczona metoda Newtona . . . 113
10.2.2. Wpływ niedokładności wyznaczenia pochodnej lub funkcji na zbieżność metody Newtona . . . 116
10.2.3. Metoda z przybliżoną pochodną . . . 116
10.2.4. Niedokładna metoda Newtona . . . 118
11. Metoda Broydena . . . 122
11.1. Realizacja metody Broydena . . . 122
11.2. Zbieżność metody Broydena . . . 124
12. Globalizacja zbieżności . . . 127
12.1. Metoda nawrotów . . . 127
12.1.1. Prosta metoda nawrotów . . . 127
12.1.2. Wielomianowe nawroty . . . 128
12.2. Metody kontynuacji . . . 129
12.2.1. Wyznaczanie dobrego przybliżenia początkowego dla metody typu Newtona . . . 130
12.2.2. Metoda homotopii . . . 131
13. Kwadratury interpolacyjne w wielu wymiarach . . . 132
13.1. Sformułowanie zadania . . . 132

13.2. Interpolacja na siatkach regularnych . . . 133
13.2.1. Postać wielomianu interpolacyjnego . . . 133
13.2.2. Błąd interpolacji . . . 135
13.3. Kwadratury interpolacyjne . . . 136
13.3.1. Kwadratury proste . . . 136
13.3.2. Kwadratury złożone . . . 137
13.4. Przekleństwo wymiaru . . . 138
14.Metody Monte Carlo . . . 140
14.1. Wstęp, metody niedeterministyczne . . . 140
14.2. Klasyczna metoda Monte Carlo . . . 140
14.2.1. Definicja i błąd . . . 140
14.2.2. Całkowanie z wagą . . . 142
14.3. Redukcja wariancji . . . 143
14.3.1. Losowanie warstwowe . . . 143
14.3.2. Funkcje kontrolne . . . 145
14.4. Generowanie liczb (pseudo-)losowych . . . 146
14.4.1. Liniowy generator kongruencyjny . . . 147
14.4.2. Odwracanie dystrybuanty i „akceptuj albo odrzuć” . . . 147
14.4.3. Metoda Box-Muller dla rozkładu gaussowskiego . . . 148
15.Metody quasi-Monte Carlo . . . 149
15.1. Co to są metody quasi-Monte Carlo? . . . 149
15.2. Dyskrepancja . . . 149
15.3. Błąd quasi-Monte Carlo . . . 151
15.3.1. Formuła Zaremby . . . 151
15.3.2. Nierówność Koksmy-Hlawki . . . 152
15.4. Ciągi o niskiej dyskrepancji . . . 154
15.4.1. Ciąg Van der Corputa . . . 154
15.4.2. Konstrukcje Haltona i Sobol’a . . . 155
15.4.3. Sieci (t, m, d) i ciągi (t, d) . . . 155
Literatura . . . 157

Wprowadzenie

Przedmiotem naszego zainteresowania będą metody rozwiązywania czterech podstawowych zadań obliczeniowych, bardzo często spotykanych w rozmaitych działach matematyki stosowanej:

Układ równań liniowych Dla danej macierzy rzeczywistej A rozmiaru $N \times N$ i wektora $b \in \mathbb{R}^N$ znaleźć $x \in \mathbb{R}^N$ taki, że

$$Ax = b.$$

Układ równań nieliniowych Dla danej funkcji $F : \mathbb{R}^N \supset D \to \mathbb{R}^N$ i wektora $b \in \mathbb{R}^N$ znaleźć $x \in D$ taki, że

$$F(x) = b.$$

Zagadnienie własne Dla danej macierzy rzeczywistej A rozmiaru $N \times N$ znaleźć $\lambda \in \mathbb{C}$ i niezerowy wektor $x \in \mathbb{C}^N$ takie, że

$$Ax = \lambda x.$$

Całkowanie Mając daną $f : \mathbb{R}^N \supset D \to \mathbb{R}$, obliczyć

$$I = \int_D f(x)\,dx.$$

Bez wielkiej przesady można powiedzieć, że w każdej poważniejszej symulacji wykonywanej na użytek inżynierski, bankowy, naukowy, itp., któreś (lub kilka) z powyższych zadań występuje jako jądro obliczeniowe, nierzadko o krytycznym znaczeniu. Dlatego jest tak ważne, by potrafić takie zadania rozwiązywać szybko i dokładnie.

Wymienione zadania mają w zastosowaniach jeszcze jedną wspólną cechę: jest nią duży (lub bardzo duży) rozmiar zadania N. Przykładowo, układy równań liniowych spotykane w symulacjach układów elektronicznych mogą mieć kilka milionów niewiadomych. W finansach konieczne jest obliczanie całek z bardzo skomplikowanych funkcji, po kostce 360-wymiarowej. Algorytm stosowany w wyszukiwarce Google musi wyznaczyć wektor własny odpowiadający dominującej wartości własnej macierzy o miliardowym rozmiarze. Dyskretyzacje nieliniowych układów równań różniczkowych (na przykład, równania Naviera–Stokesa), bardzo szybko mogą doprowadzić do nieliniowych układów równań o milionie niewiadomych, a ludzie prowadzący symulacje turbulentnego przepływu cieczy z radością uścisnęliby nas, gdybyśmy tylko dali im szansę wykonana obliczeń z 10^{12} (lub więcej!) niewiadomych.

Zadania obliczeniowe o tak wielkim rozmiarze wymagają specjalnych algorytmów numerycznych. Przykładowo, naiwnie obliczając całkę

$$\int_{[0,1]^N} f(x_1, \ldots, x_N)\,dx_1 \ldots dx_N$$

jako całkę iterowaną

$$\int_0^1 \left(\cdots \left(\int_0^1 f(x_1, \ldots, x_N)\,dx_1 \right) \ldots \right) dx_N$$

i stosując do wyznaczenia każdej z całek jednowymiarowych na przykład złożoną kwadraturę trapezów opartą na $n \geqslant 2$ węzłach, musielibyśmy obliczyć aż n^N samych tylko wartości funkcji!

(Gdy $N = 360$, jak w finansach, a za n weźmiemy skromnie $n = 10$, należałoby wyznaczyć 10^{360} samych tylko wartości funkcji — co nie rokuje zbyt dobrze, zważywszy, że współczesne komputery osobiste w ciągu **jednego roku** mogą w najlepszym razie wykonać „zaledwie" 10^{15} operacji arytmetycznych, a najszybsze na świecie superkomputery mogą teoretycznie wykonać 10^{20} takich operacji rocznie...)

W naszych rozważaniach o sposobach rozwiązywania Wielkiej Czwórki Zadań będziemy stąpać po solidnym gruncie, omawiając metody klasyczne: sprawdzone w praktyce i teoretycznie dobrze zbadane. Co ciekawe, do ich zrozumienia i analizy będzie nam potrzebna jedynie podstawowa wiedza z Analizy i GALu, a także nieco elementarnych wiadomości z Rachunku Prawdopodobieństwa. W wykładzie bardzo mało będziemy odwoływać się wprost do materiału z Matematyki Obliczeniowej I: jego treść jest w znacznej mierze *ortogonalna* do tego przedmiotu — ale oczywiście pewna wiedza o metodach numerycznych może nam pomóc lepiej umieścić omawiane kwestie na tle szerszego spektrum zagadnień matematyki obliczeniowej.

Wraz z możliwościami obliczeniowymi komputerów rosną także nasze potrzeby i wymagania, a zjawiska, jakie chcemy symulować numerycznie stają się coraz bardziej złożone, wykraczając poza podstawowe zadania omawiane w skrypcie. Mamy nadzieję, że zdobyta na niniejszym wykładzie wiedza i doświadczenie pozwolą Państwu świadomie korzystać z gotowych pakietów obliczeniowych (więcej na ten temat w wykładzie z Obliczeń naukowych) i w przyszłości zmierzyć się także z nowymi, trudniejszymi wyzwaniami.

Podstawowa literatura W części dotyczącej metod iteracyjnych dla układów równań liniowych oparliśmy się głównie na podręcznikach [13] oraz [9]. Metody iteracyjne dla układów równań nieliniowych omówiono wzorując się na [9].

Niektóre z poruszanych tu zagadnień są w ciekawy i nieco odmienny od tu zaprezentowanego sposób przedstawione w [5].

Rozdziały 1—4 oraz 13—15 napisał Leszek Plaskota, a autorem rozdziałów 5—12 jest Piotr Krzyżanowski. Dziękujemy Leszkowi Marcinkowskiemu, który prowadził wykład z tego przedmiotu, za wskazanie usterek obecnych we wcześniejszych wersjach skryptu.

1. Zagadnienie własne I

Cztery początkowe wykłady poświęcimy *zagadnieniu własnemu*. Naszym zadaniem będzie obliczenie, a raczej numeryczna aproksymacja wartości własnych danej macierzy kwadratowej (jednej, kilku, lub wszystkich). W ogólniejszym sformułowaniu możemy chcieć obliczyć również odpowiednie wektory własne.

Pierwszy wykład będzie dotyczyć przede wszystkim wrażliwości wartości i wektorów własnych na zaburzenia macierzy. Jak wiemy z podstawowego wykładu analizy numerycznej, jest to istotne z punktu widzenia numerycznej jakości algorytmów.

1.1. Podstawy teoretyczne

Zaczniemy od przypomnienia podstawowych pojęć i faktów z algebry liniowej dotyczących zagadnienia własnego, z których skorzystamy w dalszej części wykładu.

Definicja 1.1. Liczbę $\lambda \in \mathbb{C}$ (zespoloną) nazywamy *wartością własną* macierzy kwadratowej $A \in \mathbb{C}^{n,n}$ jeśli istnieje niezerowy wektor $\vec{x} \in \mathbb{C}^n$ taki, że

$$A\,\vec{x} \,=\, \lambda\,\vec{x}.$$

Wektor $\vec{x}$ nazywamy *wektorem własnym* odpowiadającym wartości własnej λ.

Równoważnie, λ jest wartością własną macierzy A gdy jest zerem jej wielomianu charakterystycznego, tzn. gdy wyznacznik

$$\det\,(A - \lambda\,I) \,=\, 0,$$

gdzie I jest macierzą identycznościową $n \times n$. *Krotnością algebraiczną* wartości własnej λ nazywamy jej krotność jako zera wielomianu charakterystycznego.

Zbiór wszystkich wektorów własnych odpowiadających danej wartości własnej λ, uzupełniony o wektor zerowy, czyli zbiór $\{\vec{x} \in \mathbb{C}^n : A\,\vec{x} = \lambda\,\vec{x}\}$, jest podprzestrzenią liniową $\mathbb{C}^n$ zwaną *podprzestrzenią własną* odpowiadającą λ. Wymiar tej podprzestrzeni to *krotność geometryczna* wartości własnej λ.

Wektory własne odpowiadające różnym wartościom własnym są liniowo niezależne.

Macierze $A, B \in \mathbb{C}^{n,n}$ są *podobne* gdy istnieje nieosobliwa macierz $C \in \mathbb{C}^{n,n}$ taka, że $B = C^{-1}\,A\,C$. Oczywiście, relacja „bycia macierzami podobnymi" jest symetryczna.

Macierze podobne mają te same wartości własne. Jeśli bowiem $A\,\vec{x} = \lambda\,\vec{x}$ to $B\,(C\,\vec{x}) = \lambda\,(C\,\vec{x})$, tzn. λ jest wartością własną B, a odpowiadający jej wektor własny to $\vec{y} = C\,\vec{x}$.

Macierz $A \in \mathbb{C}^{n,n}$ jest *diagonalizowalna* gdy jest podobna do macierzy diagonalnej, czyli gdy istnieje nieosobliwa $V = [\vec{v}_1, \ldots, \vec{v}_n] \in \mathbb{C}^{n,n}$ taka, że

$$V^{-1}\,A\,V \,=\, \Lambda \,=\, \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n).$$

Zauważmy, że wtedy możemy równoważnie zapisać $A\,V = V\,\Lambda$, albo $A\,\vec{v}_j = \lambda_j\,\vec{v}_j$ dla $1 \leqslant j \leqslant n$. Stąd elementy diagonalne macierzy Λ są wartościami własnymi macierzy A (gdzie ta sama wartość własna powtarza się tyle razy ile wynosi jej krotność), a kolumny macierzy V są odpowiednimi wektorami własnymi.

Szczególne własności mają macierze *hermitowskie* albo, w przypadku rzeczywistym, macierze *symetryczne*, bowiem dla nich istnieją bazy ortonormalne (odpowiednio w $\mathbb{C}^n$ lub $\mathbb{R}^n$) wektorów własnych. Odpowiednie twierdzenie przypominamy jedynie w przypadku rzeczywistym, który ma zasadnicze znaczenie w obliczeniach praktycznych.

Twierdzenie 1.1. *Niech A będzie macierzą symetryczną o współczynnikach rzeczywistych,*

$$A \,=\, A^T \,\in\, \mathbb{R}^{n,n}.$$

Wtedy istnieje w $\mathbb{R}^n$ baza ortonormalna wektorów własnych macierzy A, tzn. istnieje układ wektorów własnych $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n \in \mathbb{R}^n$ taki, że

$$(\vec{q}_i, \vec{q}_j)_2 \,=\, \vec{q}_j^{\,T}\, \vec{q}_i \,=\, \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$$

Oczywiście, odpowiednie wartości własne λ_j, $1 \leqslant j \leqslant n$, są też rzeczywiste.

Powyższa własność macierzy symetrycznych oznacza tyle, że są one diagonalizowalne przy pomocy macierzy *ortogonalnych*. Rzeczywiście, oznaczając

$$Q \,:=\, [\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n], \qquad \Lambda \,=\, \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

mamy $Q^T\, Q = I = Q\, Q^T$ oraz

$$Q^T\, A\, Q \,=\, \Lambda, \qquad A \,=\, Q\, \Lambda\, Q^T.$$

1.2. Teoria zaburzeń dla macierzy niesymetrycznych

Zobaczmy najpierw, czy mamy w ogóle szansę numerycznego rozwiązania zadania znalezienia wartości własnych macierzy. W tym celu zbadamy jego uwarunkowanie, czyli wrażliwość na małe względne zaburzenia współczynników macierzy.

1.2.1. Wstępny przykład

W ogólności, uwarunkowanie naszego zadania może być niestety dowolnie duże.

Przykład 1.1. Niech macierz

$$J_\varepsilon \,:=\, \begin{bmatrix} \mu & 1 & 0 & \cdots & 0 \\ 0 & \mu & 1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & \mu & 1 \\ \varepsilon & 0 & \cdots & 0 & \mu \end{bmatrix} \,\in\, \mathbb{C}^{n,n}.$$

Dla $\varepsilon = 0$, macierz $J = J_0$ jest pojedynczą klatką Jordana, której jedyną wartością własną jest μ (o krotności algebraicznej n i krotności geometrycznej 1). Jeśli teraz zaburzymy wyraz w lewym dolnym rogu o $\varepsilon > 0$ to otrzymana macierz J_ε ma już n różnych wartości własnych. Rzeczywiście, jej wielomian charakterystyczny

$$\det\,(J_\varepsilon - \lambda\, I) \,=\, (\mu - \lambda)^n + (-1)^{n+1}\varepsilon$$

ma pierwiastki

$$\mu_k = \mu + \varepsilon^{1/n} \cdot e^{2\pi\imath k/n}, \qquad 0 \leqslant k \leqslant n-1 \qquad (\imath = \sqrt{-1}).$$

Względne zaburzenie macierzy na poziomie ε powoduje więc względne zaburzenie wartości własnych na poziomie $\varepsilon^{1/n}$. Stąd, dla $n \geqslant 2$, uwarunkowanie rośnie do $+\infty$ gdy $\varepsilon \to 0^+$.

Oczywiście, otrzymane oszacowanie nie oznacza, że w powyższym przykładzie w ogóle nie potrafimy aproksymować wartości własnej. Tracimy jednak na liczbie poprawnie obliczonych cyfr znaczących wyniku. Niech $n = 2$. Wtedy przy dokładności arytmetyki 10^{-8} możemy spodziewać się jedynie wyniku z dokładnością do czterech cyfr znaczących, a przy dokładności 10^{-16} z dokładnością do ośmiu cyfr znaczących. Co więcej, im większy format n klatki Jordana tym gorzej.

Powyższy przykład pokazuje również, że praktycznie niemożliwe jest numeryczne wyznaczenie struktury macierzy. Nawet drobne zaburzenie macierzy powoduje bowiem, że pojedyncza klatka Jordana staje się macierzą diagonalizowalną.

1.2.2. Wrażliwość wartości własnych

Dalej będziemy już zakładać, że macierz A jest diagonalizowalna, czyli że jej postać Jordana jest macierzą diagonalną. Wtedy mamy następujące oszacowanie *Bauera-Fike'a*.

Twierdzenie 1.2. *Niech $A \in \mathbb{C}^{n,n}$ będzie macierzą diagonalizowalną o wartościach własnych λ_k, $1 \leqslant k \leqslant n$,*

$$A = V \Lambda V^{-1}.$$

Niech dalej μ będzie dowolną wartością własną macierzy 'sąsiedniej' $A + E$. Wtedy

$$\min_{1 \leqslant k \leqslant n} |\lambda_k - \mu| \leqslant \operatorname{cond}(V) \cdot \|E\|,$$

gdzie $\operatorname{cond}(V) = \|V\| \cdot \|V^{-1}\|$ oraz $\|\cdot\|$ jest normą macierzy indukowaną przez dowolną normę p-tą wektora.

Dowód. Załóżmy bez zmniejszenia ogólności, że μ jest różne od każdej wartości własnej λ_k. Niech $\vec{x}$ będzie wektorem własnym odpowiadającym wartości własnej μ macierzy $A+E$. Wtedy

$$V\,(\Lambda + V^{-1}\,E\,V)\,V^{-1}\,\vec{x} = \mu\,\vec{x}. \tag{1.1}$$

Oznaczając $\vec{y} = V^{-1}\,\vec{x} \neq \vec{0}$ mamy dalej

$$(\Lambda - \mu\,I)\,\vec{y} = -V^{-1}\,E\,V\,\vec{y}, \tag{1.2}$$

czyli

$$\|\vec{y}\| \leqslant \|\Lambda^{-1}\|\,\|V^{-1}\|\,\|V\|\,\|E\|\,\|\vec{y}\|.$$

Ponieważ $\Lambda^{-1} = \operatorname{diag}((\lambda_1 - \mu)^{-1}, \ldots, (\lambda_n - \mu)^{-1})$ to ostatecznie

$$\min_{1 \leqslant k \leqslant n} |\lambda_k - \mu| \leqslant \operatorname{cond}(V)\,\|E\|.$$

□

Dla macierzy diagonalizowalnych zaburzenia wartości własnych są więc lipschitzowską funkcją zaburzeń macierzy, przy czym stała Lipschitza wynosi $\mathrm{cond}(V)$.

Podane oszacowanie jest globalne dla wszystkich wartości własnych. Zaburzenia macierzy A mogą jednak w różny sposób przenosić się na zaburzenia różnych wartości własnych. Od czego to zależy? Tak jak w dowodzie twierdzenia Bauera-Fike'a, niech

$$(A+E)\,\vec{x} \;=\; \mu\,\vec{x}, \qquad \|\vec{x}\|_2^2 = \vec{x}^{\,H}\,\vec{x} = 1.$$

Biorąc jedynie i-tą współrzędną po obu stronach równania (1.2) dostajemy

$$(\lambda_i - \mu)\,\vec{z}_i^{\,H}\,\vec{x} \;=\; -\vec{z}_i^{\,H}\,E\,\vec{x}, \tag{1.3}$$

gdzie $\vec{z}_i$ jest znormalizowaną i-tą kolumną macierzy

$$(V^{-1})^H \;=\; [\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_n],$$

tzn. $\vec{z}_i = \vec{w}_i/\|\vec{w}_i\|_2$. W szczególności, $\|\vec{z}_i\|_2 = 1$ oraz

$$\vec{z}_i \;\perp\; \mathrm{span}(\vec{v}_1, \ldots, \vec{v}_{i-1}, \vec{v}_{i+1}, \ldots, \vec{v}_n)$$

(ortogonalność ze względu na zwykły iloczyn skalarny). Stąd, jeśli $\vec{z}_i$ i $\vec{x}$ nie są ortogonalne to

$$|\lambda_i - \mu| \;\leqslant\; \frac{\|E\|}{|\vec{z}_i^{\,H}\,\vec{x}|}.$$

Dla ustalenia uwagi załóżmy teraz, że minimum w twierdzeniu 1.2 jest osiągane dla $k = 1$, a podprzestrzenią własną wartości własnej λ_1 jest

$$\mathcal{V} \;=\; \mathrm{span}\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_s\},$$

gdzie wektory własne $\vec{v}_1, \ldots, \vec{v}_s$ tworzą bazę ortonormalną w $\mathcal{V}$. Jeśli wektor $\vec{x}$ jest 'dostatecznie blisko' podprzestrzeni własnej $\mathcal{V}$ (co, jak pokażemy w następnym podrozdziale, jest prawdą dla dostatecznie małego zaburzenia E) to $|\lambda_1 - \mu|$ można w przybliżeniu oszacować z góry przez maksymalną wartość wyrażenia

$$\frac{\|E\|}{\max_{1\leqslant i\leqslant s} |\vec{z}_i^{\,H}\,\vec{y}|}$$

po wszystkich $\vec{y} \in \mathcal{V}$ takich, że $\|\vec{y}\|_2 = 1$. Niech więc $\vec{y} = \sum_{i=1}^s a_i\,\vec{v}_i \in \mathcal{V}$, przy czym $\sum_{i=1}^s |a_i|^2 = 1$. Wobec tego, że dla każdego i mamy

$$\vec{z}_i^{\,H}\,\vec{y} \;=\; a_i\,\vec{z}_i^{\,H}\,\vec{v}_i \;=\; a_i/\|w_i\|_2,$$

interesujące nas 'max' jest najmniejsze gdy $a_i = \|w_i\|_2 \left(\sum_{j=1}^s \|w_j\|^2\right)^{-1/2}$. Stąd dostajemy

$$|\lambda_1 - \mu| \;\lessapprox\; \left(\sum_{i=1}^{s} \|w_i\|_2^2\right)^{1/2} \cdot \|E\| \qquad (\|E\| \to 0^+).$$

Ponieważ $\|w_i\| = 1/(\vec{z}_i^{\,H}\,\vec{v}_i)$ oraz $\vec{z}_i$ jest ortogonalny do $\mathrm{span}\{\vec{v}_j : j \neq i\}$, otrzymana nierówność sugeruje, że wartość własna λ_1 może być bardzo wrażliwa na zaburzenia macierzy gdy odpowiadająca jej podprzestrzeń własna $\mathrm{span}\{\vec{v}_1, \ldots, \vec{v}_s\}$ jest 'prawie ortogonalna' do $\mathrm{span}\{\vec{v}_{s+1}, \ldots, \vec{v}_n\}^\perp$. Dobrze ilustruje to następujący przykład.

Przykład 1.2. Rozpatrzmy macierz

$$A = \begin{bmatrix} 2 & -1/\delta \\ 0 & 1 \end{bmatrix},$$

gdzie $\delta > 0$ jest 'małe'. Wartości własne A wynoszą 1 i 2, natomiast wektory własne $[1,0]^T$ i $[1,\delta]^T$ są prawie liniowo zależne. Zaburzając macierz przez dodanie $\varepsilon = -2\delta$ do wyrazu w lewym dolnym rogu dostajemy macierz o wartościach własnych 0 i 3, a odpowiadające im wektory własne wynoszą $[1,2\delta]^T$ i $[1,-2\delta]^T$. Dodajmy, że w tym przypadku $\mathrm{cond}(V)$ jest proporcjonalne do $1/\delta$.

1.2.3. Wrażliwość wektorów własnych

Przez zaburzenie wektora własnego będziemy rozumieć odległość wektora $\vec{x}$ od podprzestrzeni własnej $\mathcal{V}$, tzn.

$$\mathrm{dist}(\vec{x},\mathcal{V}) := \min\{\, \|\vec{x}-\vec{v}\|_2 : \ \vec{v}\in\mathcal{V}\,\},$$

albo, równoważnie, długość rzutu ortogonalnego $P\vec{x}$ wektora $\vec{x}$ na podprzestrzeń

$$\mathcal{V}^\perp = \mathrm{span}(\vec{z}_{s+1},\ldots,\vec{z}_n).$$

Pokażemy, że podobnie jak przypadku wartości własnych, zaburzenie wektorów własnych jest lipszitzowską funkcją $\|E\|$. W tym celu, wybierzmy w $\mathcal{V}^\perp$ dowolną bazę ortonormalną

$$Y = [\vec{y}_{s+1},\ldots,\vec{y}_n] \in \mathbb{C}^{n,n-s}.$$

Niech $B^H \in \mathbb{C}^{n-s,n-s}$ będzie macierzą przejścia z bazy $Z = [\vec{z}_{s+1},\ldots,\vec{z}_n]$ do Y,

$$Y = Z\,B^H.$$

Wtedy

$$\|P\vec{x}\|_2 = \|Y^H\,P\vec{x}\|_2 = \|B\,Z^H\,P\vec{x}\|_2.$$

Wobec równości (1.3) mamy

$$Z^H\,P\vec{x} = -\mathrm{diag}\left((\lambda_{s+1}-\mu)^{-1},\ldots,(\lambda_n-\mu)^{-1}\right)\,B^{-1}\,Y^H\,E\,\vec{x}.$$

Ponieważ na podstawie twierdzenia Bauera-Fike'a mamy

$$|\lambda_1-\mu| \geqslant |\lambda_i-\lambda_1| - |\lambda_1-\mu| \geqslant |\lambda_i-\lambda_1| - \mathrm{cond}(V)\cdot\|E\|,$$

otrzymujemy następującą przybliżoną nierówność

$$\mathrm{dist}(\vec{x},\mathcal{V}) = \|P\vec{x}\|_2 \lessapprox \frac{\|B\|_2\|B^{-1}\|_2}{\min_{s+1\leqslant i\leqslant n}|\lambda_i-\lambda_1|}\,\|E\| \qquad (\|E\|\to 0^+). \tag{1.4}$$

Uwaga. W przykładzie 1.2 istotnemu zaburzeniu uległy wartości własne, natomiast wektory własne nie. Jest to zrozumiałe w świetle ostatniego wyniku. W przypadku macierzy 2×2 mamy bowiem $B = 1 \in \mathbb{C}^{1,1}$ i w konsekwencji wrażliwość wektorów własnych zależy tylko od różnicy $|\lambda_1-\lambda_2|$.

Poniższy przykład dla macierzy symetrycznej (!) pokazuje jak ważne dla wrażliwości wektorów własnych jest odseparowanie różnych wartości własnych.

Przykład 1.3. Wartościami własnymi macierzy symetrycznej

$$A = \begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 1 \end{bmatrix}$$

są $(1+\varepsilon)$ i 1, a odpowiadająca baza wektorów własnych to $\vec{e}_1$ i $\vec{e}_2$. Dla macierzy zaburzonej na poziomie ε (czyli różnicy wartości własnych),

$$A + E = \begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 1 \end{bmatrix} + \varepsilon \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix},$$

wartościami własnymi są $(1+\varepsilon)$ i $(1-\varepsilon)$, a bazę wektorów własnych tworzą $\vec{e}_1 + \vec{e}_2$ i $\vec{e}_1 - \vec{e}_2$. Baza wektorów własnych została obrócona o możliwie maksymalny kąt $\pi/4$.

1.3. Teoria zaburzeń dla macierzy symetrycznych

W tej części będziemy zakładać, że macierz jest rzeczywista i symetryczna,

$$A = A^T \in \mathbb{R}^{n,n}.$$

Ponieważ elementy $a_{i,j}$ i $a_{j,i}$ macierzy A są w praktycznych obliczeniach reprezentowane przez tą samą zmienną, zasadne jest założenie, że macierz zaburzona $A+E$ jest również symetryczna, $E = E^T \in \mathbb{R}^{n,n}$.

1.3.1. Wrażliwość wartości własnych

Przypomnijmy twierdzenie 1.1, które mówi, że dla macierzy symetrycznej istnieje baza ortonormalna Q jej wektorów własnych. Ponieważ dla macierzy ortogonanych mamy $\|Q\|_2 = 1$,

$$\mathrm{cond}_2(Q) = \|Q\|_2 \|Q^T\|_2 = 1.$$

To zaś, razem z twierdzeniem 1.2 implikuje, że każda watość własna μ macierzy sąsiedniej $A+E$ spełnia nierówność

$$\min_{1 \leqslant i \leqslant n} |\lambda_i - \mu| \leqslant \|E\|_2.$$

Możemy jednak pokazać dużo więcej. Zachodzi bowiem następujące twierdzenie Weyla.

Twierdzenie 1.3. *Niech $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$ będą wartościami własnymi macierzy $A = A^T \in \mathbb{R}^{n,n}$, a $\mu_1 \geqslant \mu_2 \geqslant \cdots \geqslant \mu_n$ wartościami własnymi macierzy sąsiedniej $A + E$, gdzie $E = E^T \in \mathbb{R}^{n,n}$. Wtedy dla wszystkich $1 \leqslant k \leqslant n$ mamy*

$$|\lambda_k - \mu_k| \leqslant \|E\|_2.$$

Dowód twierdzenia opiera się na następującej pożytecznej nierówności.

Lemat 1.1. *Dla dowolnej rzeczywistej macierzy symetrycznej A mamy*

$$\max_{\dim(\mathcal{V})=k} \min_{\vec{0} \neq \vec{x} \in \mathcal{V}} \frac{\vec{x}^T A \vec{x}}{\vec{x}^T \vec{x}} = \lambda_k = \min_{\dim(\mathcal{W})=n-k+1} \max_{\vec{0} \neq \vec{y} \in \mathcal{W}} \frac{\vec{x}^T A \vec{x}}{\vec{x}^T \vec{x}}, \tag{1.5}$$

przy czym odpowiednie maksimum i minimum osiągane są dla $\mathcal{V}^* = \mathrm{span}(\vec{v}_k, \vec{v}_{k+1}, \ldots, \vec{v}_n)$ oraz $\mathcal{W}^* = \mathrm{span}(\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k)$.

Dowód. Niech $\mathcal{V}$ i $\mathcal{W}$ będą dowolnymi podprzestrzeniami o wskazanych wymiarach. Ponieważ suma wymiarów wynosi $n+1$ to istnieje wektor niezerowy $\vec{z} \in \mathcal{V} \cap \mathcal{W}$. W konsekwencji

$$\min_{\vec{0}\neq\vec{x}\in\mathcal{V}} \frac{\vec{x}^T A\,\vec{x}}{\vec{x}^T\,\vec{x}} \leqslant \frac{\vec{z}^T A\,\vec{z}}{\vec{z}^T\,\vec{z}} \leqslant \max_{\vec{0}\neq\vec{y}\in\mathcal{W}} \frac{\vec{x}^T A\,\vec{x}}{\vec{x}^T\,\vec{x}}.$$

Biorąc maximum po $\mathcal{V}$ i minimum po $\mathcal{W}$ dostajemy, że w (1.5) lewa strona nie jest większa od prawej. Aby pokazać odwrotną nierówność, zauważmy, że dla $\mathcal{V}^*$ i $\mathcal{W}^*$ mamy odpowiednio

$$\min_{\vec{0}\neq\vec{x}\in\mathcal{V}^*} \frac{\vec{x}^T A\,\vec{x}}{\vec{x}^T\,\vec{x}} \geqslant \min_{\vec{0}\neq\sum_{j\geqslant k} a_j\,\vec{v}_j} \frac{\sum_{j\geqslant k} a_j^2\lambda_j}{\sum_{j\geqslant k} a_j^2} = \lambda_k,$$

$$\max_{\vec{0}\neq\vec{y}\in\mathcal{W}^*} \frac{\vec{y}^T A\,\vec{y}}{\vec{y}^T\,\vec{y}} \leqslant \max_{\vec{0}\neq\sum_{j\leqslant k} b_j\,\vec{v}_j} \frac{\sum_{j\leqslant k} b_j^2\lambda_j}{\sum_{j\leqslant k} b_j^2} = \lambda_k.$$

□

Dla dowodu twierdzenia 1.3 zastosujemy lemat 1.5 najpierw do macierzy $A+E$, a potem do macierzy $A = (A+E) - E$. Otrzymujemy odpowiednio

$$\begin{aligned} \mu_k &= \min_{\dim(\mathcal{W})=n-k+1} \max_{\vec{0}\neq\vec{y}\in\mathcal{W}} \frac{\vec{x}^T (A+E)\,\vec{x}}{\vec{x}^T\,\vec{x}} \\ &\leqslant \min_{\dim(\mathcal{W})=n-k+1} \max_{\vec{0}\neq\vec{y}\in\mathcal{W}} \frac{\vec{x}^T A\,\vec{x}}{\vec{x}^T\,\vec{x}} + \|E\|_2 \\ &= \lambda_k + \|E\|_2, \end{aligned}$$

oraz podobnie nierówność odwrotną $\lambda_k \leqslant \mu_k + \|E\|_2$, czyli ostatecznie

$$|\lambda_k - \mu_k| \leqslant \|E\|_2.$$

Zanotujmy jeszcze, że wielkość

$$\frac{\vec{x}^T A\,\vec{x}}{\vec{x}^T\,\vec{x}}$$

znana jest pod nazwą *ilorazu Rayleigh'a*.

1.3.2. Wrażliwość wektorów własnych

Niech $\mathcal{V}_k \subset \mathbb{R}^n$ będzie podprzestrzenią własną odpowiadającą wartości własnej λ_k macierzy A, a $\vec{x}_k$ wektorem własnym odpowiadającym wartości własnej μ_k macierzy sąsiedniej $A+E$. Niech dalej θ_k będzie kątem pomiędzy $\vec{x}_k$ i podprzestrzenią $\mathcal{V}_k$. Wtedy

$$\sin\theta_k \lessapprox \frac{\|E\|_2}{\min_{\lambda_j\neq\lambda_k} |\lambda_k - \lambda_j|} \qquad (\|E\|_2 \to 0^+).$$

Rzeczywiście, to wynika bezpośrednio z twierdzenia 1.3, nierówności (1.4) oraz faktu, że macierz B w (1.4) jest identycznością. Pokażemy teraz nierówność dokładną.

Twierdzenie 1.4.

$$\frac{1}{2}\sin 2\theta_k \leqslant \frac{\|E\|_2}{\min_{\lambda_j\neq\lambda_k} |\lambda_k - \lambda_j|}.$$

Dowód. Dla ustalenia uwagi załóżmy, że $k = 1$ oraz odpowiednia podprzestrzeń własna $\mathcal{V}_1 = \mathrm{span}(\vec{v}_1, \ldots, \vec{v}_s)$. Przedstawmy wektor własny $\vec{x}_1$, $\|\vec{x}\|_2 = 1$, w jednoznaczny sposób w postaci

$$\vec{x}_1 \,=\, \vec{v} + \vec{d},$$

gdzie $\vec{v} \in \mathcal{V}_1$ i $\vec{d} \in \mathcal{V}_1^{\perp}$,

$$\vec{d} \,=\, \sum_{j=s+1}^{n} b_j \, \vec{v}_j.$$

Możemy też założyć, bez straty ogólności, że $\vec{d} \neq \vec{0}$ i $\vec{x}_1 \notin \mathcal{V}_1^{\perp}$, tzn. $0 < \theta < \pi/2$.

Przekształcając równanie $(A + E)\,(\vec{v} + \vec{d}) = \mu_1\,(\vec{v} + \vec{d})$ dostajemy

$$\vec{z} \,:=\, (A - \lambda_1\, I)\,\vec{d} \,=\, ((\mu_1 - \lambda_1)\, I - E)\;(\vec{v} + \vec{d}).$$

Ponieważ każdy z wektorów $\vec{v}_i$ dla $1 \leqslant i \leqslant s$ należy do jądra macierzy symetrycznej $A - \lambda_1\, I$ to wektor $(A - \lambda_1\, I)\,\vec{d}$ jest ortogonalny do $\mathcal{V}_1$ i tym samym możemy zapisać

$$\vec{z} \,=\, \sum_{j=s+1}^{n} a_j \, \vec{v}_j.$$

Mamy dalej

$$(A - \lambda_1\, I)\,\vec{d} = \,(A - \lambda_1\, I)\;\left(\sum_{j=s+1}^{n} b_j \, \vec{v}_j\right) \,=\, \sum_{j=s+1}^{n} (\lambda_j - \lambda_1) b_j \, \vec{v}_j,$$

a stąd $a_j = (\lambda_j - \lambda_1) b_j$ i

$$\|\vec{d}\|_2 \,=\, \left(\sum_{j=s+1}^{n} \frac{a_j^2}{(\lambda_j - \lambda_1)^2}\right)^{1/2} \,\leqslant\, \frac{\|\vec{z}\|_2}{\min_{s+1 \leqslant j \leqslant n} |\lambda_j - \lambda_1|}.$$

Z kolei z równości $\vec{v}^{\,T}\;((\mu_1 - \lambda_1)\, I - E)\;(\vec{v} + \vec{d}) = 0$ dostajemy

$$(\mu_1 - \lambda_1)\,\|\vec{v}\|_2^2 \,=\, \vec{v}^{\,T}\, E\,(\vec{v} + \vec{d}),$$

skąd

$$\vec{z} \,=\, \frac{1}{\|\vec{v}\|_2^2}(\vec{v} + \vec{d})\,\vec{v}^{\,T}\, E\,(\vec{v} + \vec{d}) - E\,(\vec{v} + \vec{d}) \,=\, \left(\frac{1}{\|\vec{v}\|_2^2}(\vec{v} + \vec{d})\,\vec{v}^{\,T} - I\right)\, E\,(\vec{v} + \vec{d}).$$

Gdybyśmy teraz wiedzieli, że

$$\left\| \frac{1}{\|\vec{v}\|_2^2}(\vec{v} + \vec{d})\,\vec{v}^{\,T} - I \right\|_2 \,=\, \frac{1}{\|\vec{v}\|_2} \tag{1.6}$$

to moglibyśmy napisać

$$\|\vec{z}\|_2 \,\leqslant\, \frac{\|\vec{v} + \vec{d}\|_2^2}{\|\vec{v}\|_2}\|E\|_2 \,=\, \frac{\|E\|_2}{\|\vec{v}\|_2}$$

i ostatecznie

$$\frac{1}{2}\sin 2\theta_1 \,=\, \sin\theta_1 \cos\theta_1 \,=\, \|\vec{d}\|_2\|\vec{v}\|_2 \,\leqslant\, \frac{\|\vec{z}\|_2\|\vec{v}\|_2}{\min_{s+1 \leqslant j \leqslant n} |\lambda_j - \lambda_1|} \,\leqslant\, \frac{\|E\|_2}{\min_{s+1 \leqslant j \leqslant n} |\lambda_j - \lambda_1|}.$$

Pozostaje więc do pokazania równość (1.6). W tym celu, weźmy $\vec{y} = \alpha\,\vec{v}/\|\vec{v}\|_2 + \beta\,\vec{h}$, gdzie $\|h\|_2 = 1$, $\vec{h} \perp \vec{v}$ i $\|\vec{y}\|_2 = (\alpha^2+\beta^2)^{1/2} = 1$. Wtedy

$$\frac{(\vec{v}+\vec{d})\,\vec{v}^{\,T}}{\|\vec{v}\|_2^2}\,\vec{y} - \vec{y} \;=\; \frac{\alpha\,\vec{d}}{\|\vec{v}\|_2} - \beta\,\vec{h}.$$

Dla danych α i β, wektor po prawej stronie ma największą normę gdy $\vec{h} = \pm\vec{d}/\|\vec{d}\|_2$, przy czym bierzemy plus wtedy i tylko wtedy gdy α i β mają różne znaki. Dlatego poszukiwana norma wynosi

$$\max_{\alpha^2+\beta^2=1} \alpha \cdot \frac{\|\vec{d}\|_2}{\|\vec{v}\|_2} + \beta.$$

Zamieniając zmienne na $\alpha = \cos\phi$, $\beta = \sin\phi$ łatwo dostajemy, że optymalne $\phi \in (0, \pi/2)$ spełnia $\sin\phi = \|\vec{d}\|_2$, $\cos\phi = \|\vec{v}\|_2$, a stąd dostajemy wynik

$$\frac{\|\vec{d}\|_2}{\|\vec{v}\|_2}\,\|\vec{d}\|_2 + \|\vec{v}\|_2 \;=\; \frac{\|\vec{d}\|_2^2 + \|\vec{v}\|_2^2}{\vec{v}\|_2} \;=\; \frac{1}{\|\vec{v}\|_2}.$$

□

Pokazaliśmy, że wektory własne są mało wrażliwe na zaburzenia macierzy o ile wartości własne są odseparowane od siebie na poziomie dużo większym niż $\|E\|_2$. W szczególności, jeśli dodatkowo $\lambda_i \neq \lambda_j$, $i \neq j$, to mamy jednolite oszacowanie

$$\frac{1}{2}\,\sin 2\theta_k \;\leqslant\; \frac{\|E\|_2}{\min_{i\neq j}|\lambda_i - \lambda_j|}.$$

Przypomnijmy jeszcze przykład 1.3 pokazujący, że warunek odseparowania wartości własnych jest konieczny.

2. Zagadnienie własne II

Kolejne dwa wykłady poświęcimy konkretnym metodom numerycznym rozwiązywania zagadnienia własnego.

2.1. Uwagi wstępne

Ponieważ wartości własne macierzy A są zerami jej wielomianu charakterystycznego, narzucająca się metoda poszukiwania wartości własnych polegałaby na wyliczeniu współczynników tego wielomianu, na przykład w bazie potęgowej, a następnie na zastosowaniu jednej ze znanych metod znajdowania zer wielomianu. Dla wielomianów o współczynnikach rzeczywistych mogłaby to być np. metoda bisekcji, metoda Newtona (siecznych), albo pewna kombinacja obu method. Należy jednak przestrzec przed dokładnie takim postępowaniem. Rzecz w tym, że zera wielomianu mogą być bardzo wrażliwe na zaburzenia jego współczynnikówi. Dobrze ilustruje to następujący przykład.

Przykład 2.1. Załóżmy, że A jest macierzą symetryczną formatu 20×20 o wartościach własnych $1, 2, \ldots, 20$. Zapisując wielomian charakterystyczny w postaci potęgowej mamy

$$\det(A - \lambda\, I) \;=\; \prod_{k=1}^{20} (\lambda_k - k) \;=\; \sum_{k=0}^{20} w_k \lambda^k.$$

Okazuje się, że zaburzenie współczynnika w_{19} mogą powodować 10^{10} razy większe zaburzenie wartości własnej $\lambda_{16} = 16$.

Nie znaczy to oczywiście, że metody bazujące na obliczaniu wielomianu charakterystycznego, czy jego pochodnych trzeba z gruntu odrzucić. Trzeba tylko pamiętać, aby przy obliczeniach korzystać bezpośrednio ze współczynników $a_{i,j}$ macierzy A, ponieważ, jak wiemy z poprzedniego rozdziału, zadanie jest ze względu na te współczynniki dobrze uwarunkowane.

Z drugiej strony zauważmy, że policzenie wyznacznika macierzy pełnej wprost z definicji jest raczej kosztowne. Dlatego w praktyce metody wyznacznikowe są zwykle poprzedzone prekomputingiem polegającym na sprowadzeniu macierzy przez podobieństwa ortogonalne do prostszej postaci, z której wyznacznik można już obliczyć dużo tańszym kosztem niż dla macierzy pełnej. Tego typu precomputing ma również zastosowanie w innych metodach, w którym np. trzeba wykonywać mnożenie macierzy przez wektor.

Jedną z takich wygodnych postaci macierzy jest postać Hessenberga, a dla macierzy hermitowskich postać trójdiagonalna.

2.1.1. Sprowadzanie macierzy do postaci Hessenberga

Macierz $A = (a_{i,j}) \in \mathbb{C}^{n,n}$ jest postaci *Hessenberga* („prawie" trójkątną górną) jeśli wszystkie wyrazy $a_{i,j}$ dla $i \geqslant j+2$ są zerami, tzn.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & a_{3,2} & \cdots & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix}.$$

Oczywiście, jeśli macierz jest hermitowska, $A = A^H$, to postać Hessenberga jest równoważna postaci trójdiagonalnej

$$A = \begin{bmatrix} c_1 & b_2 & 0 & \cdots & 0 \\ \overline{b}_2 & c_2 & b_3 & \cdots & 0 \\ 0 & \overline{b}_3 & c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \cdots & c_n \end{bmatrix}. \tag{2.1}$$

Aby daną macierz A sprowadzić do postaci Hessenberga przez podobieństwa ortogonalne (a więc nie zmieniając wartości własnych), możemy zastosować znane nam odbicia Householdera. Przypomnijmy, że dla dowolnego niezerowego wektora $\vec{a} \in \mathbb{C}^n$ istnieje macierz ortogonalna (odbicie Householdera) $P = P^H = P^{-1} \in \mathbb{C}^{n,n}$ postaci $P = I - 2\,\vec{u}\,\vec{u}^{\,H}$, gdzie $\|\vec{u}\|_2 = 1$, która przekształca $\vec{a}$ na kierunek pierwszego wersora, tzn. $P\,\vec{a} = \ell\,\vec{e}_1$, $|\ell| = \|\vec{a}\|_2$. (Odbicia Householdera zostały dokładnie przedstawione na wykładzie *Analiza Numeryczna I.*)

Niech $A = (a_{i,j}) \in \mathbb{C}^{n,n}$ będzie dowolną macierzą. Algorytm konstruuje najpierw odbicie $P_1 \in \mathbb{C}^{n-1,n-1}$ dla wektora $[a_{2,1}, a_{3,1}, \ldots, a_{n,1}]^T$, a następnie biorąc macierz

$$\widehat{P}_1 = \begin{bmatrix} 1 & \vec{0}^{\,T} \\ \vec{0} & P_1 \end{bmatrix} \in \mathbb{C}^{n,n}$$

oblicza $A^{(1)} = \widehat{P}_1\, A\, \widehat{P}_1^H$. łatwo zobaczyć, że wtedy elementy $(i,1)$ dla $i = 3, 4, \ldots, n$ w macierzy $A^{(1)}$ są równe zeru.

Postępując indukcyjnie załóżmy, że dostaliśmy już macierz $A^{(k)} = (a_{i,j}^{(k)})$, $1 \leqslant k \leqslant n-3$, w której elementy $a_{i,j}^{(k)}$ dla $i \geqslant j+2$, $1 \leqslant j \leqslant k$, są wyzerowane. W kroku $(k+1)$-szym algorytm konstruuje odbicie $P_{k+1} \in \mathbb{C}^{n-k,n-k}$ dla wektora $[a_{k+2,k+1}^{(k)}, \ldots, a_{n,k+1}^{(k)}]^T$, a następnie biorąc macierz

$$\widehat{P}_{k+1} = \begin{bmatrix} I_k & 0^T \\ 0 & P_{k+1} \end{bmatrix} \in \mathbb{C}^{n,n}$$

oblicza $A^{(k+1)} = \widehat{P}_{k+1}\, A^{(k)}\, \widehat{P}_{k+1}^H$. Wtedy wszystkie elementy $a_{i,j}^{(k+1)}$ dla $i \geqslant j+2$, $1 \leqslant j \leqslant k+1$, są zerami. Po $(n-2)$ krokach otrzymana macierz $\widetilde{A} = A^{(n-1)}$ jest postaci Hessenberga.

Jasne jest, że jeśli $A = A^H$ to opisany algorytm prowadzi do macierzy trójdiagonalnej, $\widetilde{A} = \widetilde{A}^H$. (Obliczenia można wtedy w każdym kroku wykonywać jedynie na głównej diagonali i pod nią.)

Dodajmy jeszcze, że koszt sprowadzenia macierzy przez podobieństwa ortogonalne do postaci Hessenberga/trójdiagonalnej jest proporcjonalny do n^3.

2.2. Metoda Hymana

Przy pomocy metody *Hymana* możemy w zręczny sposób obliczyć wartości i pochodne wielomianu charakterystycznego $\det(A - \lambda I)$ dla macierzy Hessenberga $A = (a_{i,j}) \in \mathbb{C}^{n,n}$. Bez zmniejszenia ogólności będziemy zakładać, że wszystkie elementy $a_{i+1,i} \neq 0$. W przeciwnym przypadku wyznacznik można obliczyć jako iloczyn wyznaczników macierzy Hessenberga niższych rzędów.

Metoda Hymana polega na dodaniu do pierwszego wiersza macierzy $A - \lambda I$ wiersza i-tego pomnożonego przez pewien współczynnik $q_i = q_i(\lambda)$, dla $i = 2, 3, \ldots, n$ tak, aby wyzerować elementy $(1, i)$ dla $i = 1, 2, \ldots, n-1$. Ponieważ

$$A - \lambda I = \begin{bmatrix} a_{1,1} - \lambda & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} - \lambda & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & a_{3,2} & \cdots & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a_{n,n-1} & a_{n,n} - \lambda \end{bmatrix},$$

mamy następujące równania na q_i:

$$\begin{aligned} (a_{1,1} - \lambda) + a_{1,2}q_2 &= 0, \\ a_{1,1} + a_{2,1}q_2 + \cdots + a_{i-1,i}q_{i-1} + (a_{i,i} - \lambda)q_i + a_{i+1,i}q_{i+1} &= 0, \end{aligned} \tag{2.2}$$

dla $i = 2, 3, \ldots, n-1$. Stąd, definiując dodatkowo $q_1 = 1$, dostajemy następujące równania rekurencyjne:

$$\begin{aligned} q_2(\lambda) &= \frac{-(a_{1,1} - \lambda)}{a_{2,1}}, \\ q_{i+1}(\lambda) &= \frac{-\left(a_{1,i}q_1 + \cdots + a_{i-1,i}q_{i-1} + (a_{i,i} - \lambda)q_i\right)}{a_{i+1,i}}. \end{aligned}$$

Po opisanym przekształceniu macierzy $A - \lambda I$ zmieni się jedynie jej pierwszy wiersz; wyrazy $a_{1,i}$, $1 \leqslant i \leqslant n-1$, zostaną wyzerowane, a $a_{1,n}$ zostanie przekształcony do

$$\widehat{a}_{1,n} = a_{1,n}q_1 + \cdots + a_{n-1,n}q_{n-1} + (a_{n,n} - \lambda)q_n.$$

Rozwijając wyznacznik względem (przekształconego) pierwszego wiersza otrzymujemy

$$\det(A - \lambda I) = (-1)^{n+1} a_{2,1}a_{3,2} \cdots a_{n,n-1} \left(a_{1,n}q_1 + \cdots + a_{n-1,n}q_{n-1} + (a_{n,n} - \lambda)q_n\right),$$

a stąd zera wielomianu charakterystycznego są równe zerom wielomianu

$$q_{n+1}(\lambda) = -\left(a_{1,n}q_1 + \cdots + a_{n-1,n}q_{n-1} + (a_{n,n} - \lambda)q_n\right).$$

Oczywiście, wartości $q_{n+1}(\lambda)$ można obliczać stosując wzory rekurencyjne (2.2) przedłużając je o $i = n$, przy dodatkowym formalnym podstawieniu $a_{n+1,n} = 1$.

Aby móc zastowoać netodę Newtona (stycznych) do znalezienia zer wielomianu q_{n+1} potrzebujemy również wiedzieć jak obliczać jego pochodne. To też nie jest problem, bo rekurencyjne wzory na pochodne można uzyskać po prostu różniczkując wzory (2.2). Dodajmy, że koszt obliczenia wartości $q_{n+1}(\lambda)$ i $q'_{n+1}(\lambda)$ jest proporjonalny do n^2, co jest istotnym zyskiem w porównaniu z kosztem n^3 obliczania wyznacznika pełnej macierzy za pomocą faktoryzacji metodą eliminacji Gaussa.

Formuły na obliczanie wyznacznika $\det(A - \lambda I)$ i jego pochodnych uproszczają się jeszcze bardziej gdy macierz A jest dodatkowo hermitowska, czyli jest macierzą trójdiagonalną postaci (2.1). Wtedy, stosując zwykłe rozumowanie rekurencyjne, łatwo się przekonać, że kolejne minory główne (czyli wyznaczniki macierzy kątowych) spełniają zależności

$$\begin{aligned} p_0(\lambda) &= 1, \qquad p_1(\lambda) = c_1 - \lambda, \\ p_i(\lambda) &= (c_i - \lambda)p_{i-1}(\lambda) - |b_i|^2 p_{i-2}(\lambda) \qquad \text{dla} \quad i = 2, 3, \ldots, n. \end{aligned}$$

Różniczkując otrzymane wzory otrzymujemy formuły na pochodne kolejnych minorów po λ. Wartości wielomianu $\det(A - \lambda I) = p_n(\lambda)$ oraz jego pochodnych można więc obliczać kosztem liniowym w n.

Sprawę konkretnej implementacji metod iteracyjnych bisekcji i/lub Newtona do wyznaczenia zer wielomianu $\det(A - \lambda I)$ tutaj pomijamy. Ograniczymy się jedynie do stwierdzenia, że nie jest to rzecz całkiem trywialna.

2.3. Metoda potęgowa

2.3.1. Definicja metody

Metoda potęgowa zdefiniowana jest w następujący prosty sposób. Rozpoczynając od dowolnego wektora $\vec{x}_0 \in \mathbb{C}^n$ o normie $\|\vec{x}_0\|_2 = 1$ obliczamy kolejno dla $k = 0, 1, 2, \ldots$

$$\vec{y}_k := A\,\vec{x}_k, \qquad \vec{x}_{k+1} := \frac{\vec{y}_k}{\|\vec{y}_k\|_2}.$$

Równoważnie możemy napisać

$$\vec{x}_k = \frac{A^k\,\vec{x}_0}{\|A^k\,\vec{x}_0\|_2}.$$

Wektory $\vec{x}_k$ stanowią kolejne przybliżenia wektora własnego. Odpowiadającą mu wartość własną przybliżamy wzorem

$$\eta_k := \vec{x}_k^H\,A\,\vec{x}_k = \vec{x}_k^H\,\vec{y}_k.$$

2.3.2. Analiza zbieżności

Analizę metody potęgowej przeprowadzimy przy założeniu, że macierz A jest diagonalizowalna. Oznaczmy przez μ_i, $1 \leqslant i \leqslant s$, różne wartości własne macierzy A, a przez $\mathcal{V}_i$ odpowiadające im podprzestrzenie własne,

$$\mathcal{V}_1 \oplus \mathcal{V}_2 \oplus \cdots \oplus \mathcal{V}_s = \mathbb{C}^n. \tag{2.3}$$

Założymy, że μ_1 jest dominującą wartością własną oraz

$$|\mu_1| > |\mu_2| > \cdots > |\mu_s|.$$

Przedstawmy $\vec{x}_0$ jednoznacznie w postaci

$$\vec{x}_0 = \sum_{j=1}^{s} \vec{v}_j$$

gdzie $v_j \in \mathcal{V}_j$. Dla uproszczenia, będziemy zakładać, że każda ze składowych

$$\vec{v}_j \neq \vec{0}.$$

Podkreślmy, że nie jest to założenie ograniczające, bo w przeciwnym przypadku składowe zerowe po prostu ignorujemy w poniższej analizie zbieżności. Poza tym, jeśli wektor początkowy jest wybrany losowo, to teoretyczne prawdopodobieństwo takiego zdarzenia jest zerowe. Co więcej, nawet jeśli jedna ze składowych znika to wskutek błędów zaokrągleń w procesie obliczeniowym składowa ta w wektorze $\vec{x}_1$ będzie z pewnością niezerowa. (Mamy tu do czynienia z ciekawym zjawiskiem, kiedy błędy zaokrągleń pomagają!)

Prosty rachunek pokazuje, że

$$A^k \vec{x}_0 \;=\; \sum_{j=1}^{s} A^k \vec{v}_j \;=\; \sum_{j=1}^{s} \mu_j^k \vec{v}_j \;=\; \mu_1^k \left(\vec{v}_1 + \sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^k \vec{v}_j \right).$$

Ponieważ każdy z ilorazów $\mu_j/\mu_1 < 1$ to $\vec{x}_k$ zbiega do wektora własnego $\vec{v}_1/\|\vec{v}_1\|_2$, a η_k do dominującej wartości własnej μ_1. Przyjrzyjmy się od czego zależy szybkość zbieżności.

Odległość $\mathrm{dist}(\vec{x}_k, \mathcal{V}_1)$ wektora $\vec{x}_k$ od podprzestrzeni $\mathcal{V}_1$ można oszacować z góry przez odległość $\vec{x}_k$ od $\mathrm{span}(\vec{v}_1)$, która z kolei jest równa długości rzutu $P_1\vec{x}_k$ tego wektora na podprzestrzeń ortogonalną do $\vec{v}_1$,

$$P_1\vec{x}_k \;=\; \vec{x}_k \;-\; \frac{\vec{v}_1^{\,H}\,\vec{x}_k}{\|\vec{v}_1\|_2^2}\,\vec{v}_1.$$

Oznaczając

$$\beta_k \;:=\; \left\| \vec{v}_1 + \sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^k \vec{v}_j \right\|_2$$

mamy $\lim_{k\to\infty} \beta_k = \|\vec{v}_1\|_2$ oraz

$$\begin{aligned}
P_1\vec{x}_k \;&=\; \frac{1}{\beta_k} \left(\left(\vec{v}_1 + \sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^k \vec{v}_j \right) - \left(\|\vec{v}_1\|_2^2 + \sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^k \vec{v}_1^{\,H}\,\vec{v}_j \right) \frac{\vec{v}_1}{\|\vec{v}_1\|_2^2} \right) \\
&=\; \frac{1}{\beta_k} \left(\sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^k \left(\vec{v}_j - \frac{\vec{v}_1^{\,H}\,\vec{v}_j}{\|\vec{v}_1\|_2^2}\,\vec{v}_1 \right) \right) \\
&\approx\; \left(\frac{\mu_j}{\mu_1} \right)^k \frac{1}{\|\vec{v}_1\|_2} \left(\vec{v}_2 - \frac{\vec{v}_1^{\,H}\,\vec{v}_2}{\|\vec{v}_1\|_2^2}\,\vec{v}_1 \right) \qquad\qquad (k \to \infty).
\end{aligned}$$

Biorąc normę i stosując nierówność trójkąta dostajemy

$$\mathrm{dist}(\vec{x}_k, \mathcal{V}_1) \;=\; \|P_1\vec{x}_k\|_2 \;\lessapprox\; 2 \cdot \rho_1^k \cdot \frac{\|\vec{v}_2\|_2}{\|\vec{v}_1\|_2} \qquad\qquad (k \to \infty),$$

gdzie

$$\rho_1 \;:=\; \frac{|\mu_2|}{|\mu_1|} \;<\; 1.$$

Zbieżność jest więc liniowa z ilorazem ρ_1.

Zobaczmy teraz jak bardzo η_k różni się od μ_1. Mamy

$$\begin{aligned}
\eta_k \;&=\; \vec{x}_k^{\,H}\,A\,\vec{x}_k \;=\; \frac{\mu_1}{\beta^2} \left(\vec{v}_1^{\,H} + \sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^k \vec{v}_j^{\,H} \right) \left(\vec{v}_1 + \sum_{j=2}^{s} \left(\frac{\mu_j}{\mu_1} \right)^{k+1} \vec{v}_1^{\,H}\,\vec{v}_j \right) \\
&=\; \frac{\mu_1}{\beta^2} \left(\|\vec{v}_1\|_2^2 + \sum_{j=2}^{2} \left(\frac{\mu_j}{\mu_1} \right)^k \left(\vec{v}_j^{\,H}\,\vec{v}_1 + \frac{\mu_j}{\mu_1}\,\vec{v}_1^{\,H}\,\vec{v}_j \right) + \sum_{i,j=2}^{s} \left(\frac{\mu_i^k \mu_j^{k+1}}{\mu_1^{2k+1}} \right) \vec{v}_i^{\,H}\,\vec{v}_j \right).
\end{aligned}$$

Stąd wynika, że błąd $|\mu_1 - \eta_k|$ zależy od iloczynów skalarnych $\vec{v}_j^{\,H}\,\vec{v}_1$ oraz od stosunków

$$\rho_j := \frac{|\mu_{j+1}|}{|\mu_1|}, \qquad 1 \leqslant j \leqslant s-1.$$

Dokładniej, niech

$$\ell := \min\left\{2 \leqslant j \leqslant s:\ \vec{v}_j^{\,H}\,\vec{v}_1 \neq 0\right\}$$

albo $\ell = s+1$ jeśli powyższy zbiór jest pusty. Jeśli teraz $\ell = s+1$ albo mamy jednocześnie $2 \leqslant \ell \leqslant s$ i $\rho_\ell < \rho_1^2$ to

$$|\mu_1 - \eta_k| \lessapprox |\mu_1| \cdot \rho_1^{2k+1} \cdot \frac{\|\vec{v}_2\|_2}{\|\vec{v}_1\|_2}$$

i zbieżność jest liniowa z ilorazem ρ_1^2. Jeśli zaś $2 \leqslant \ell \leqslant s$ i $\rho_\ell \geqslant \rho_1^2$ to

$$|\mu_1 - \eta_k| \lessapprox |\mu_1| \cdot \rho_\ell^{k} \cdot \frac{\|\vec{v}_2\|_2}{\|\vec{v}_1\|_2}$$

i zbieżność jest liniowa z ilorazem ρ_ℓ. W szczególności, jeśli $\vec{v}_2^{\,H}\,\vec{v}_1 \neq 0$ to zbieżość jest z ilorazem ρ_1.

Dla macierzy rzeczywistych i symetrycznych możemy stosunkowo łatwo pokazać dokładne nierówności na błąd metody potęgowej.

Twierdzenie 2.1. *Załóżmy, że macierz $A = A^T \in \mathbb{R}^{n,n}$. Niech γ_k będzie kątem pomiędzy wektorem k-tego przybliżenia $\vec{x}_k$ otrzymanego metodą potęgową i podprzestrzenią własną $\mathcal{V}_1$ odpowiadającą dominującej wartości własnej μ_1. Wtedy*

$$\tan\gamma_k \leqslant \rho_1 \cdot \tan\gamma_{k-1}$$

oraz

$$|\mu_1 - \eta_k| \leqslant \max_{2\leqslant j\leqslant s} |\mu_1 - \mu_j| \cdot \sin^2\gamma_k.$$

Dowód. Ponieważ macierz jest symetryczna, podprzestrzenie własne $\mathcal{V}_j$ zdefiniowane w (2.3) są parami ortogonalne. Dlatego

$$\vec{x}_k = \frac{A^k\,\vec{x}_0}{\|A^k\,\vec{x}_0\|_2} = \frac{\vec{v}_1 + \sum_{j=2}^s \left|\frac{\mu_j}{\mu_1}\right|^k \vec{v}_j}{\sqrt{\|\vec{v}_1\|_2^2 + \sum_{j=2}^s \left|\frac{\mu_j}{\mu_1}\right|^{2k} \|\vec{v}_j\|_2^2}}.$$

Tangens kąta γ_k jest równy stosunkowi długości składowej wektora $\vec{x}_k$ w podprzestrzeni $\mathcal{V}_2 \oplus \cdots \oplus \mathcal{V}_s$ do długości składowej tego wektora w podprzestrzeni $\mathcal{V}_1$. Mamy więc

$$\tan\gamma_k = \left(\sum_{j=2}^s \left|\frac{\mu_j}{\mu_1}\right|^{2k} \frac{\|\vec{v}_j\|_2^2}{\|\vec{v}_1\|_2^2}\right)^{1/2} \leqslant \max_{2\leqslant j\leqslant s}\left|\frac{\mu_j}{\mu_1}\right| \cdot \tan\gamma_{k-1}.$$

Aby pokazać pozostałą część twierdzenia, przedstawmy $\vec{x}_k$ w postaci $\vec{x}_k = \sum_{j=1}^s \vec{z}_j$, gdzie $\vec{z}_j \in \mathcal{V}_j$. Wtedy $\sum_{j=1}^s \|\vec{z}_j\|_2^2 = \|\vec{x}_k\|_2^2 = 1$ oraz

$$\vec{x}_k^{\,T}\,A\,\vec{x}_k = \sum_{j=1}^s \mu_j \|\vec{z}_j\|_2^2 = \mu_1 + \sum_{j=2}^s (\mu_j - \mu_1)\,\|\vec{z}_j\|_2^2,$$

a stąd

$$|\mu_1 - \eta_k| \leqslant \max_{2\leqslant j\leqslant s} |\mu_1 - \mu_j| \cdot \sum_{j=2}^{s} \|\vec{z}_j\|_2^2 = \max_{2\leqslant j\leqslant s} |\mu_1 - \mu_j| \cdot \sin^2 \gamma_k.$$

□

Przypomnijmy, że $\sin\gamma_k \leqslant \tan\gamma_k$ przy czym mamy asymptotyczną równość gdy $\gamma_k \to 0$. Ponieważ macierz A jest symetryczna to mamy również $|\mu_i - \mu_j| \leqslant 2\|A\|_2$. Z twierdzenia 2.1 wynika więc, że

$$\tan\gamma_k \leqslant \rho_1^k \cdot \tan\gamma_0, \qquad \text{oraz} \qquad |\mu_1 - \eta_k| \leqslant 2\rho_1^{2k} \cdot \|A\|_2 \cdot \tan^2\gamma_0.$$

Metoda potęgowa nie opłaca się gdy wymiar n nie jest duży, albo macierz A jest pełna. Inaczej jest, gdy n jest „wielkie", np. rzędu co najmniej kilkuset, a macierz A jest rozrzedzona, tzn. ma jedynie proporcjonalnie do n elementów niezerowych. Z taką sytuacją mamy do czynienia, gdy np. macierz jest pięciodiagonalna. Wtedy istotne dla metody mnożenie $A\,\vec{x}$ można wykonać kosztem proporcjonalnym do n i poświęcając tyle samo pamięci (wyrazy zerowe można zignorować).

2.4. Odwrotna metoda potęgowa i deflacja

Odwrotna metoda potęgowa albo *metoda Wielandta*, polega na zastosowaniu (prostej) metody potęgowej do macierzy $(A - \sigma I)^{-1}$. Dokładniej, startując z przybliżenia początkowego $\vec{x}_0$ o jednostkowej normie drugiej obliczamy kolejno dla $k = 0, 1, 2, \ldots$

$$\vec{y}_k := (A - \sigma I)^{-1}\,\vec{x}_k, \qquad \vec{x}_{k+1} := \frac{\vec{y}_k}{\|\vec{y}_k\|_2},$$

oraz $\eta_k := \vec{x}_k^H\, A\,\vec{x}_k$.

Od razu nasuwają się dwie uwagi. Po pierwsze, iteracja odwrotna ma sens tylko wtedy gdy parametr σ jest tak dobrany, że macierz $A - \sigma I$ jest nieosobliwa, co jest równoważne warunkowi $\sigma \neq \mu_i$ dla wszystkich i. Po drugie, w realizacji numerycznej, dla znalezienia wektora $\vec{y}_k$ nie odwracamy macierzy $(A - \sigma I)$, ale rozwiązujemy układ równań $(A - \sigma I)\,\vec{y} = \vec{x}_k$, co czyni metodę tak samo kosztowną co iteracja prosta.

Analiza iteracji odwrotnych przebiega podobnie do analizy iteracji prostych dla macierzy $(A - \sigma I)^{-1}$. Nietrudno zauważyć, że macierz ta ma te same wektory własne co A, a jej wartości własne wynoszą

$$\mu_i^{(\sigma)} = \frac{1}{\mu_i - \sigma}, \qquad 1 \leqslant i \leqslant s.$$

(To wynika bezpośrednio z równości $(A + \sigma I)\,\vec{v}_i = (\mu_i + \sigma)\,\vec{v}_i$.) Dlatego iteracje odwrotne zbiegają do wektora własnego odpowiadającego wartości własnej μ_{i^*} takiej, że

$$|\mu_{i^*} - \sigma| = \min_{1\leqslant i\leqslant s} |\mu_i - \sigma|,$$

przy czym szybkość zbieżności zależy teraz od wielkości

$$\rho_j^{(\sigma)} := \frac{|\mu_{i^*} - \sigma|}{\min_{i\neq i^*} |\mu_i - \sigma|}$$

(a nie od $\rho_j = |\mu_{j+1}|/|\mu_1|$, jak w iteracji prostej).

Niewątpliwą zalety odwrotnej metody potęgowej jest to, że zbiega do wartości własnej najbliższej σ, przy czym im σ bliższe μ_{i^*} tym lepiej. Metoda jest więc szczególnie efektywna w przypadku gdy znamy dobre przybliżenia wartości własnych macierzy A. Niestety, taka informacja nie zawsze jest dana wprost.

Pamiętając, że kolejne przybliżenia wartości własnej w metodzie potęgowej są obliczane przy pomocy ilorazów Rayleigha, dobrym pomysłem na przesunięcie wydaje się być wzięcie w k-tym kroku iteracji odwrotnej

$$\sigma = \eta_k = \vec{x}_k^H \, A \, \vec{x}_k.$$

Rzeczywiście. Niech γ_k będzie kątem pomiędzy $\vec{x}_k$ a podprzestrzenią własną V_{i^*}. Zakładając, że η_k jest dostatecznie blisko μ_{i^*} oraz przyjmując $\delta = \min_{i \neq i^*} |\mu_{i^*} - \mu_i|$, na podstawie twierdzenia 2.1 mamy

$$\begin{aligned} \tan \gamma_{k+1} &\leqslant \frac{|\mu_{i^*} - \eta_k|}{\min_{i \neq i^*} |\mu_i - \eta_k|} \cdot \tan \gamma_k \leqslant \frac{|\mu_{i^*} - \eta_k|}{\delta - |\mu_{i^*} - \eta_k|} \cdot \tan \gamma_k \\ &\leqslant \frac{2 \, \|A\|_2 \tan^3 \gamma_k}{\delta - 2\|A\|_2 \tan^2 \gamma_k} \approx \frac{2}{\delta} \cdot \|A\|_2 \tan^3 \gamma_k. \end{aligned}$$

Zamiast zbieżności kwadratowej, jak w prostej metodzie potęgowej, dostaliśmy (asymptotycznie!) zbieżność sześcienną.

Metoda potęgowa, prosta lub odwrotna, pozwala wyznaczyć tylko jedną wartość własną, powiedzmy μ_1, oraz odpowiadający jej wektor własny. Naturalne jest teraz pytanie o to jak postępować, aby znaleźć inne pary własne. Jednym ze sposobów jest zastosowanie *deflacji*.

Jeśli macierz jest hermitowska, $A = A^H$, oraz znaleźliśmy wektor własny $\vec{v}_1$ odpowiadający wartości własnej μ_1, to możemy ponowić proces metody potęgowej ograniczając go do podprzestrzeni prostopadłej do $\vec{v}_1$ wymiaru $n - 1$. Osiągamy to startując z wektora $\vec{x}_0$ prostopadłego do $\vec{v}_1$, tzn.

$$\vec{x}_0 := \frac{\vec{v}}{\|\vec{v}\|_2}, \qquad \vec{v} := \vec{w} - \vec{v}_1 \, (\vec{v}_1^H \, \vec{w}),$$

gdzie $\vec{w} \neq \vec{0}$ jest wybrany losowo. Przy idealnej realizacji procesu konstruowany ciąg $\{\vec{x}_k\}$ należałby do podprzestrzeni prostopadłej do $\vec{v}_1$. W obecności błędów zaokrągleń należałoby to wymusić poprzez reortogonalizację,

$$\vec{y}_k := \vec{y}_k - \vec{v}_1 \, (\vec{v}_1^H \, \vec{y}_k),$$

wykonywaną np. co kilka kroków. Po znalezieniu drugiej pary własnej, proces deflacji możemy kontynuować, ograniczając się do odpowiedniej podprzestrzeni własnej wymiaru $n - 2$.

3. Zagadnienie własne III

Metodę potęgową można traktować jako iteracje podprzestrzeni jednowymiarowej startujące ze span($\vec{x}_0$). Naturalnym uogólnieniem tego procesu są iteracje podprzestrzeni o wyższych wymiarach. Właśnie iteracje podprzestrzeni są punktem wyjścia konstrukcji obecnie najpopularniejszego algorytmu QR, którego zadaniem jest obliczenie jednocześnie wszystkich wartości własnych.

Chociaż metody prezentowane w tym rozdziale mogą być stosowane w większej ogólności, dla uproszczenia będziemy zakładać, że macierz A jest nieosobliwa, rzeczywista i symetryczna, tzn. $A = A^T \in \mathbb{R}^{n,n}$ i $\det(A) \neq 0$. Przypomnijmy, że wtedy istnieje baza ortonormalna $\{\vec{\xi}_i\}_{i=1}^n$ wektorów własnych macierzy,

$$A\,\vec{\xi}_i = \lambda_i\,\vec{\xi}_i, \quad 1 \leqslant i \leqslant n.$$

Będziemy również zakładać, że wartości własne są uporządkowane tak, że

$$|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n| > 0.$$

3.1. Iteracje podprzestrzeni

3.1.1. Algorytm ogólny

Niech $1 \leqslant p \leqslant n-1$ oraz $\mathcal{Y}_0 \subseteq \mathbb{R}^n$ będzie podprzestrzenią o wymiarze p. Dla $k = 1, 2, 3, \ldots$ rozpatrzmy następujące iteracje:

$$\mathcal{Y}_k \;:=\; A(\mathcal{Y}_{k-1}) \;=\; \{\, A\,\vec{x} :\; \vec{x} \in \mathcal{Y}_{k-1} \,\}.$$

Oczywiście, wobec nieosobliwości A wszystkie podprzestrzenie $\mathcal{Y}_k$ mają też wymiar p. Pokażemy, że przy pewnych niekrępujących założeniach ciąg podprzestrzeni $\mathcal{Y}_k$ zbiega do podprzestrzeni własnej

$$\mathcal{P}^{(p)} := \operatorname{span}(\vec{\xi}_1, \vec{\xi}_2, \ldots, \vec{\xi}_p),$$

przy czym przez „zbieżność" rozumiemy tu zbieżność do zera kąta pomiędzy tymi podprzestrzeniami.

Przypomnijmy, że kąt pomiędzy dwiema podprzestrzeniami $\mathcal{Y}, \mathcal{Z} \subseteq \mathbb{R}^n$ definiujemy jako

$$\angle(\mathcal{Y}_k, \mathcal{P}^{(p)}) := \max_{\vec{y} \in \mathcal{Y}_k} \min_{\vec{z} \in \mathcal{P}^{(p)}} \angle(\vec{y}, \vec{z}).$$

(Jako ćwiczenie można wykazać, że jeśli $\dim(\mathcal{Y}) = \dim(\mathcal{Z})$ to $\angle(\mathcal{Y}, \mathcal{Z}) = \angle(\mathcal{Z}, \mathcal{Y})$.)

Twierdzenie 3.1. *Niech*

$$\rho_p := \frac{|\lambda_{p+1}|}{|\lambda_p|} < 1 \qquad \textit{oraz} \qquad \gamma_k := \angle(\mathcal{Y}_k, \mathcal{P}^{(p)}), \quad k = 0, 1, 2, \ldots.$$

Jeśli $\gamma_0 < \pi/2$ *to*

$$\operatorname{tg}\gamma_k \;\leqslant\; \rho_p \cdot \operatorname{tg}\gamma_{k-1} \;\leqslant\; \rho_p^k \cdot \operatorname{tg}\gamma_0.$$

Dowód. Niech $\vec{y}$ będzie wektorem spełniającym $\angle(\vec{y}, \mathcal{P}^{(p)}) < \pi/2$. Wtedy

$$\vec{y} = \sum_{i=1}^{n} \vec{v}_i, \qquad \vec{v}_i \in \operatorname{span}(\vec{\xi}_i), \quad 1 \leqslant i \leqslant n,$$

gdzie nie wszystkie $\vec{v}_i$ dla $1 \leqslant i \leqslant p$, są są zerowe, oraz $A\,\vec{y} = \sum_{i=1}^{n} \lambda_i \vec{v}_i$. Stąd

$$\operatorname{tg} \angle(A\,\vec{y}, \mathcal{P}^{(p)}) = \sqrt{\frac{\sum_{i=p+1}^{n} |\lambda_i|^2 \|\vec{v}_i\|_2^2}{\sum_{i=1}^{p} |\lambda_i|^2 \|\vec{v}_i\|_2^2}} \leqslant \frac{|\lambda_{p+1}|}{|\lambda_p|} \cdot \sqrt{\frac{\sum_{i=p+1}^{n} \|\vec{v}_i\|_2^2}{\sum_{i=1}^{p} \|\vec{v}_i\|_2^2}} = \rho_p \cdot \operatorname{tg} \angle(\vec{y}, \mathcal{P}^{(p)} p). \quad (3.1)$$

Jeśli teraz $\gamma_{k+1} = \angle(\vec{y}_{k+1}, \mathcal{P}^{(p)})$ dla $\vec{y}_{k+1} \in \mathcal{Y}_{k+1}$ oraz $\vec{y}_{k+1} = A\,\vec{y}_k$ to z (3.1) wynika, że

$$\operatorname{tg} \gamma_{k+1} \leqslant \rho_p \cdot \operatorname{tg} \angle(\vec{y}_k, \mathcal{P}^{(p)}) \leqslant \rho_p \cdot \operatorname{tg} \gamma_k,$$

co kończy dowód. □

3.1.2. Iteracje ortogonalne

W praktyce, iteracje podprzestrzeni realizujemy reprezentując kolejne $\mathcal{Y}_k$ przez ich bazy ortonormalne. Wtedy algorytm, zwany w tym przypadku również iteracjami ortogonalnymi, przybiera następującą postać. Wybieramy macierz kolumnowo-ortogonalną Z_0 formatu $n \times p$ jako macierz startową, a następnie dla $k = 1, 2, 3, \ldots$ iterujemy:

$$\text{(IO)} \quad \begin{cases} Y_k \;:=\; A\,Z_{k-1}; \\ Y_k \;:=\; Z_k\,R_k; \quad \text{(ortonormalizacja)} \end{cases}$$

przy czym w drugiej linii następuje ortonormalizacja macierzy Y_k realizowana poprzez rozkład tej macierzy na iloczyn macierzy kolumnowo-ortonormalnej Z_k formatu $n \times p$ i macierzy trójkątnej górnej R_k formatu $p \times p$. Dokonuje się tego znanym nam już algorytmem wykorzystującym odbicia Householdera.

łatwo zauważyć, że jeśli przyjmiemy, iż wektory-kolumny macierzy Z_0 tworzą bazę ortonormalną podprzestrzeni $\mathcal{Y}_0$ to w kolejnych krokach wektory-kolumny Z_k tworzą bazę ortonormalną podprzestrzeni $\mathcal{Y}_k$. Rzeczywiście, ponieważ $Y_k = A\,Z_{k-1}$ to kolumny Y_k stanowią bazę $\mathcal{Y}_k$. Kolumny te są z kolei liniową kombinacją kolumn Z_k, czyli rozpinają tą samą podprzestrzeń co kolumny Z_k.

Poczynimy teraz kluczową obserwację. Przypuśćmy, że wszystkie wartości własne λ_i są parami różne. Gdybyśmy rozpoczynali iteracje od macierzy $\overline{Y}_0$ składającej się jedynie z $\overline{p}$ początkowych kolumn macierzy Y_0, przy czym $1 \leqslant \overline{p} < p$, to otrzymane w procesie iteracyjnym macierze $\overline{Y}_k$ i $\overline{Z}_k$ (formatu $n \times \overline{p}$ i $\overline{p} \times \overline{p}$) byłyby odpowiednio „okrojonymi" macierzami Y_k i Z_k. Stąd, jeśli wszystkie wartości własne macierzy A mają różne moduły, to dla każdego takiego $\overline{p}$ podprzestrzeń $\mathcal{Y}_k^{(\overline{p})}$ rozpięta na początkowych $\overline{p}$ wektorach macierzy Z_k „zbiega" do podprzestrzeni własnej $\mathcal{P}^{(\overline{p})}$, przy czym

$$\operatorname{tg} \angle \left(\mathcal{Y}_k^{(\overline{p})}, \mathcal{P}^{(\overline{p})} \right) \leqslant \rho_{\overline{p}}^{k} \cdot \operatorname{tg} \angle \left(\mathcal{Y}_0^{(\overline{p})}, \mathcal{P}^{\overline{p}} \right).$$

Z powyższej obserwacji wynika, że gdybyśmy realizowali iteracje podprzestrzeni przyjmując $p = n$ i startując z macierzy identycznościowej $Z_0 := I$ formatu $n \times n$ to mielibyśmy zbieżność $\mathcal{Y}_k^{(p)}$ do podprzestrzeni własnych $\mathcal{P}^{(p)}$ dla wszystkich $p = 1, 2, \ldots, n$.

W przypadku gdy pracujemy na bazach ortogonalnych i startujemy z $Z_0 = I$, zbieżność tą można wyrazić również w następujący sposób. Oznaczmy

$$V = [\vec{\xi}_1, \vec{\xi}_2, \ldots, \vec{\xi}_n], \qquad Z_k = [\vec{z}_1^{(k)}, \vec{z}_2^{(k)}, \ldots, \vec{z}_n^{(k)}].$$

Lemat 3.1. *Niech B_k będzie macierzą przejścia od bazy $(\vec{\xi}_1, \ldots, \vec{\xi}_n)$ do $(\vec{z}_1^{(k)}, \ldots, \vec{z}_n^{(k)})$,*

$$Z_k = V\, B_k.$$

Dla $1 \leqslant p \leqslant n-1$, niech

$$B_k = \begin{bmatrix} B_{1,1}^{(k)} & B_{1,2}^{(k)} \\ B_{2,1}^{(k)} & B_{2,2}^{(k)} \end{bmatrix},$$

gdzie $B_{1,1}^{(k)}$ jest podmacierzą formatu $p \times p$, a pozostałe podmacierze odpowiednio formatów $p \times (n-p)$, $(n-p) \times p$, $(n-p) \times (n-p)$. Jeśli $\rho_p = |\lambda_{p+1}|/|\lambda_p| < 1$ i $\gamma_0 < \pi/2$ to

$$\|B_{1,2}^{(k)}\|_2, \|B_{2,1}^{(k)}\|_2 \leqslant \rho_p^k \cdot \operatorname{tg} \gamma_0.$$

Dowód. Niech $Z_k = [Z_1^{(k)}, Z_2^{(k)}]$, $V = [V_1, V_2]$, gdzie $Z_1^{(k)}$ i V_1 są formatu $n \times p$. Wtedy

$$Z_1^{(k)} = V_1\, B_{1,2}^{(k)} + V_2\, B_{2,1}^{(k)}.$$

Mnożąc to równanie z lewej strony przez V_2^T dostajemy $B_{2,1}^{(k)} = V_2^T\, Z_1^{(k)}$. Stąd

$$\|B_{2,1}^{(k)}\|_2 \;=\; \|V_2^T\, Z_1^{(k)}\|_2 \;=\; \sup_{\|\vec{x}\|_2=1} \|V_2^T\, (Z_1^{(k)}\, \vec{x})\|_2.$$

Oznaczając $\vec{y} = Z_1^{(k)}\, \vec{x}$ zauważamy, że $\vec{y} \in \mathcal{Y}_k := \operatorname{span}(\vec{z}_1^{(k)}, \ldots, \vec{z}_p^{(k)})$ i $\|\vec{y}\|_2 = 1$. Stąd wektor $V_2^T\, \vec{y}$ zawiera współczynniki rzutu ortogonalnego $\vec{y}$ na $(\mathcal{P}^{(p)})^\perp := \operatorname{span}(\vec{\xi}_{p+1}, \ldots, \vec{\xi}_n)$ w bazie $(\vec{\xi}_{p+1}, \ldots, \vec{\xi}_n)$. Z definicji normy drugiej i twierdzenia 3.1 dostajemy więc

$$\|B_{2,1}^{(k)}\|_2 = \sup_{\|\vec{x}\|_2=1} \sin \angle \left(Z_1^{(k)}\, \vec{x}, (\mathcal{P}^{(p)})^\perp \right) \leqslant \rho_p^k \cdot \operatorname{tg} \gamma_0.$$

Rozumując analogicznie i wykorzystując fakt, że

$$\sin \angle(\mathcal{Y}_k, \mathcal{P}^{(p)}) = \sin \angle \left(\mathcal{Y}_k^\perp, (\mathcal{P}^{(p)})^\perp \right)$$

dostajemy taką samą nierówność dla $\|B_{1,2}^{(k)}\|_2$. □

Z lematu 3.1 wynika natychmiast, że jeśli moduły wartości własnych macierzy A są parami różne to wyrazy pozadiagonalne $b_{i,j}^{(k)}$ macierzy B_k zbiegają do zera, a ponieważ B_k jest ortogonalna, to $|b_{i,i}^{(k)}|$ zbiegają do jedynki. Algorytm (IO) można więc uzupełnić o obliczanie w każdym kroku macierzy

$$A_k \;:=\; Z_k^T\, A\, Z_k,$$

która powinna zbiegać do $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Fakt ten, łącznie z analizą szybkości zbieżności, pokażemy formalnie w następnym podrozdziale.

3.2. Metoda QR

3.2.1. Wyprowadzenie metody

Załóżmy, że realizujemy algorytm iteracji ortogonalnych (IO) z macierzą początkową $Z_0 = I$. Wtedy $A\, Z_{k-1} = Z_k\, R_k$, skąd

$$A_{k-1} = Z_{k-1}^T\, A\, Z_{k-1} = (Z_{k-1}^T\, Z_k)\, R_k.$$

Oznaczając $Q_k := Z_{k-1}^T\, Z_k$ widzimy, że ostatnia równość to nic innego jak rozkład ortogonalno-trójkątny macierzy A_{k-1},

$$A_{k-1} = Q_k\, R_k.$$

Prawdziwe są więc związki rekurencyjne

$$\begin{aligned} A_k &= Z_k^T\, A\, Z_k = (Z_k^T\, Z_{k-1})\,(Z_{k-1}^T\, A\, Z_{k-1})\,(Z_{k-1}^T\, Z_k) \\ &= Q_k^T\, A_{k-1}\, Q_k, \\ Z_k &= Z_{k-1}\,(Z_{k-1}^T\, Z_k) = Z_{k-1}\, Q_k. \end{aligned}$$

Zależności te prowadzą do *ALGORYTMU QR*, który oblicza kolejne macierze A_k i Z_k w następujący sposób.

$$\text{(QR)} \qquad \begin{array}{l} A_0 := A; \quad Z_0 := I; \\ \text{dla} \quad k = 1, 2, 3, \ldots \\ \left\{ \begin{array}{l} A_{k-1} := Q_k\, R_k; \quad \text{(ortonormalizacja)}, \\ A_k := R_k\, Q_k; \quad Z_k := Z_{k-1}\, Q_k; \end{array} \right. \end{array}$$

3.2.2. QR a iteracje ortogonalne

Przypomnijmy, że w rozkładzie ortogonalno-trójkątnym macierz ortogonalna jest wyznaczona jednoznacznie z dokładnością do znaków jej wektorów-kolumn. Dlatego macierze Z_k (a tym samym także A_k) powstałe w wyniku realizacji algorytmów (IO) i (QR) nie muszą być takie same. Algorytmy te są jednak równoważne w następującym sensie.

Lemat 3.2. *Niech $\widetilde{Z}_k$, $\widetilde{A}_k$ i Z_k, A_k będą macierzami powstałymi w wyniku realizacji odpowiednio algorytmów (IO) i (QR). Wtedy*

$$\widetilde{Z}_k \;=\; Z_k\, D_k \qquad \textit{oraz} \qquad \widetilde{A}_k = D_k\, A_k\, D_k,$$

gdzie D_k są pewnymi macierzami diagonalnymi z elementami ± 1 na głównej przekątnej.

Dowód. Zastosujemy indukcję względem k. Dla $k = 1$ mamy $\widetilde{Z}_0 = I = Z_0$ oraz $\widetilde{A}_0 = A = A_0$. Załóżmy, że lemat jest prawdziwy dla $k-1$, tzn. $\widetilde{Z}_{k-1} = Z_k\, D_{k-1}$. Wtedy, z jednej strony

$$\widetilde{A}_{k-1} \;=\; D_{k-1}\, A_{k-1}\, D_{k-1} \;=\; D_{k-1}\, Q_k\, R_k\, D_{k-1},$$

a z drugiej strony

$$\widetilde{A}_{k-1} \;=\; \widetilde{Z}_{k-1}^T\, \widetilde{Z}_k\, \widetilde{R}_k \;=\; D_{k-1}\, Z_{k-1}^T\, \widetilde{Z}_k\, \widetilde{R}_k.$$

Mamy więc

$$D_{k-1}\, \widetilde{A}_{k-1} \;=\; (Z_{k-1}^T\, \widetilde{Z}_k)\, \widetilde{R}_k \;=\; Q_k\, (R_k\, D_{k-1}),$$

przy czym równości te przedstawiają dwa rozkłady ortogonalno-trójkątne tej samej macierzy $D_{k-1}\, \widetilde{A}_{k-1}$. A jeśli tak, to czynniki ortogonalne tych rozkładów różnią się jedynie znakami kolumn. Równoważnie, istnieje macierz diagonalna D_k z elementami ± 1 na głównej przekątnej taka, że

$$Z_{k-1}^T\, \widetilde{Z}_k \;=\; Q\, D_k \qquad (\text{oraz} \quad \widetilde{R}_k = D_k\, R_k\, D_{k-1}).$$

Stąd

$$\widetilde{Z}_k \;=\; Z_{k-1}\, Q_k\, D_k \;=\; Z_k\, D_k.$$

Dowód kończy następujący rachunek:

$$\widetilde{A}_k \;=\; \widetilde{Z}_k^T\, A\, \widetilde{Z}_k \;=\; D_k^T\, (Z_k^T\, A\, Z_k)\, D_k \;=\; D_k\, A_k\, D_k.$$

□

Zauważmy jeszcze, że jeśli przez $\widetilde{a}_{i,j}^{(k)}$ i $a_{i,j}^{(k)}$ oznaczymy odpowiednio elementy macierzy $\widetilde{A}_k$ i A_k oraz $D_k = \operatorname{diag}(d_1^{(k)}, \ldots, d_n^{(k)})$ z $d_i^{(k)} = \pm 1$ to

$$\widetilde{a}_{i,j}^{(k)} = d_i^{(k)} d_j^{(k)} a_{i,j}^{(k)}.$$

To oznacza, że dla $i \neq j$ elementy te różnią się jedynie znakiem, a dla $i = j$ są sobie równe.

3.2.3. Analiza zbieżności

Jesteśmy już gotowi do przedstawienia twierdzeń o zbieżności metody QR.

Twierdzenie 3.2. *Dla $1 \leqslant p \leqslant n$, niech $\rho_p := |\lambda_{p+1}/\lambda_p| < 1$ i $\gamma_k := \angle\left(\mathcal{Y}_k, \mathcal{P}^{(p)}\right)$. Niech dalej*

$$A_k = \begin{bmatrix} A_{1,1}^{(k)} & A_{1,2}^{(k)} \\ A_{2,1}^{(k)} & A_{2,2}^{(k)} \end{bmatrix}, \quad Q_k = \begin{bmatrix} Q_{1,1}^{(k)} & Q_{1,2}^{(k)} \\ Q_{2,1}^{(k)} & Q_{2,2}^{(k)} \end{bmatrix}, \quad R_k = \begin{bmatrix} R_{1,1}^{(k)} & R_{1,2}^{(k)} \\ 0 & R_{2,2}^{(k)} \end{bmatrix},$$

gdzie podmacierze $A_{1,1}^{(k)}, Q_{1,1}^{(k)}, R_{1,1}^{(k)}$ są formatu $p \times p$, a pozostałe podmacierze formatów odpowiednio $p \times (n-p)$, $(n-p) \times p$ i $(n-p) \times (n-p)$. Wtedy, przyjmując

$$\varepsilon_p^{(k)} = \rho_p^k \cdot \operatorname{tg} \gamma_0$$

mamy $\operatorname{tg} \gamma_k \leqslant \varepsilon_p^{(k)}$ oraz

$$\|A_{1,2}^{(k)}\|_2 = \|A_{2,1}^{(k)}\|_2 \leqslant 2\,\varepsilon_p^{(k)} \|A\|_2, \tag{3.2}$$

$$\|Q_{1,2}^{(k)}\|_2,\ \|Q_{2,1}^{(k)}\|_2 \leqslant 2\,\varepsilon_p^{(k)}, \tag{3.3}$$

$$\|R_{1,2}^{(k)}\|_2 \leqslant 4\,\varepsilon_p^{(k)} \|A\|_2. \tag{3.4}$$

Dowód. Wobec wykazanej w lemacie 3.2 (teoretycznej) równoważności metody QR i iteracji (IO), nierówność $\operatorname{tg} \gamma_k \leqslant \varepsilon_p^{(k)}$ została już udowodniona w twierdzeniu 3.1. Aby pokazać (3.3), zapiszemy Z_k w postaci $Z_k = [Z_1^{(k)}, Z_2^{(k)}]$, gdzie $Z_1^{(k)}$ i $Z_2^{(k)}$ składają się odpowiednio z pierwszych p i z ostatnich $(n-p)$ kolumn macierzy Z_k. Wtedy

$$Q_{2,1} = Z_2^{(k-1)\,T} Z_1^{(k)} = \left(V^T Z_2^{(k-1)}\right)^T \left(V^T Z_1^{(k)}\right),$$

gdzie $V = [\vec{\xi}_1, \vec{\xi}_2, \ldots, \vec{\xi}_n]$. Pisząc $V = [V_1, V_2]$, podobnie jak dla Z_k, mamy teraz

$$V^T Z_1^{(k)} = \begin{bmatrix} V_1^T Z_1^{(k)} \\ V_2^T Z_1^{(k)} \end{bmatrix} =: \begin{bmatrix} F \\ G \end{bmatrix}.$$

Ponieważ $\|V_1\|_2 = \|Z_1^{(k)}\|_2 = 1$ to $\|F\|_2 \leqslant 1$, natomiast $\|G\|_2 \leqslant \varepsilon_p^{(k)}$, jak pokazaliśmy w dowodzie lematu 3.1.

Pisząc z kolei

$$V^T Z_2^{(k-1)} =: \begin{bmatrix} F' \\ G' \end{bmatrix},$$

w analogiczny sposób pokazujemy, że $\|F'\|_2 \leqslant 1$ i $\|G'\|_2 \leqslant \varepsilon_p^{(k)}$. Stąd mamy

$$\|Q_{2,1}^{(k)}\|_2 = \|G^T F + F^T G\|_2 \leqslant \|G'\|_2\|F\| + \|F'\|_2\|G\|_2 \leqslant \varepsilon_{k-1} + \varepsilon_p^{(k)} \leqslant 2 \cdot \varepsilon_p^{(k)}.$$

Dokładnie tak samo pokazujemy oszacowanie dla $\|Q_{1,2}\|_2$.

Aby pokazać (3.2) zauważmy, że wobec $A_k = Q_k R_k$ mamy też $A_{2,1}^{(k)} = Q_{2,1}^{(k)} R_{1,1}^{(k)}$. Ale

$$\|R_{1,1}\|_2 \leqslant \|R_k\|_2 = \|A_2\|_2 = \|A\|_2,$$

co w połączeniu z (3.3) dowodzi (3.2).

W końcu, wobec $R_k = A_k Q_k^T$ mamy

$$R_{1,2}^{(k)} = A_{1,1}^{(k)} {Q_{2,1}^{(k)}}^T + A_{1,2}^{(k)} {Q_{2,2}^{(k)}}^T.$$

Nierówność (3.4) wynika teraz z nierówności $\|A_{1,1}^{(k)}\|_2 \leqslant \|A_k\|_2 = \|A\|_2$, (3.2), (3.3), oraz $\|Q_{2,2}^{(k)}\|_2 \leqslant 1$. □

Z twierdzenia 3.2 wynika, że elementy pozadiagonalne macierzy A_k zbiegają do zera liniowo, przy czym iloraz zbieżności zależy od wartości $\rho_p = |\lambda_{p+1}|/|\lambda_p|$. Dokładniej, dla danego elementu $a_{i,j}^{(k)}$ macierzy A_k, gdzie $1 \leqslant i < j \leqslant n$, prawdziwe jest oszacowanie

$$|a_{i,j}^{(k)}| \leqslant \min\left(\varepsilon_i^{(k)}, \varepsilon_{i+1}^{(k)}, \ldots \varepsilon_{j-1}^{(k)}\right)$$

(gdzie skorzystaliśmy także z faktu, że moduł pojedynczego elementu macierzy jest nie większy od normy drugiej tej macierzy).

A jaka jest zbieżność elementów diagonalnych A_k do wartości własnych macierzy A? Oznaczmy

$$\overline{\varepsilon}_p^{(k)} = \begin{cases} \varepsilon_1^{(k)} & p = 1, \\ \max\left(\varepsilon_{p-1}^{(k)}, \varepsilon_p^{(k)}\right) & 2 \leqslant p \leqslant n-1, \\ \varepsilon_{n-1}^{(k)} & p = n. \end{cases}$$

Twierdzenie 3.3. *Dla wszystkich $1 \leqslant p \leqslant n$ mamy*

$$|a_{p,p}^{(k)} - \lambda_p| \leqslant 4 \cdot \left(\overline{\varepsilon}_p^{(k)}\right)^2 \cdot \|A\|_2.$$

Dowód. Wobec $Z_k = V B_k$ mamy

$$\begin{aligned} a_{p,p}^{(k)} &= (\vec{z}_p^{(k)})^T A \, \vec{z}_p^{(k)} = \left(V \vec{b}_p^{(k)}\right)^T A \left(V \vec{b}_p^{(k)}\right) \\ &= \vec{b}_p^{(k)} V^T A V \vec{b}_p^{(k)} = \vec{b}_p^{(k)} \Lambda \vec{b}_p^{(k)} \\ &= \lambda_p \left(b_{p,p}^{(k)}\right)^2 + \sum_{i \neq p} \lambda_i \left(b_{i,j}^{(k)}\right)^2. \end{aligned}$$

Stąd i z lematu 3.1 otrzymujemy

$$\begin{aligned} |a_{p,p}^{(k)} - \lambda_p| &= \left|-\lambda_p \left(1 - \left(b_{p,p}^{(k)}\right)^2\right) + \sum_{i \neq p} \lambda_i \left(b_{i,p}^{(k)}\right)^2\right| = \left|\sum_{i \neq p} \left(b_{i,p}^{(k)}\right)^2 (\lambda_i - \lambda_p)\right| \\ &\leqslant 2 \cdot \|A\|_2 \cdot \left(\sum_{i<p} \left(b_{i,p}^{(k)}\right)^2 + \sum_{p<i} \left(b_{i,p}^{(k)}\right)^2\right) \leqslant 4 \cdot \left(\overline{\varepsilon}_p^{(k)}\right)^2 \cdot \|A\|_2. \end{aligned}$$

□

Elementy diagonalne $a^{(k)}_{p,p}$ macierzy A_k zbiegają więc do wartości własnych λ_p macierzy A co najmniej jak $|\lambda_2/\lambda_1|^{2k}$ dla $p=1$, $|\lambda_n/\lambda_{n-1}|^{2k}$ dla $p=n$, oraz $\min\left\{|\lambda_p/\lambda_{p-1}|^{2k}, |\lambda_{p+1}/\lambda_p|^{2k}\right\}$ dla $2 \leqslant p \leqslant n-1$.

3.3. QR z przesunięciami i deflacja

Tak jak w przypadku metody potęgowej, można spróbować przyspieszyć zbieżność metody QR poprzez przesunięcie widma macierzy A, czyli zastosowanie QR do macierzy $\overline{A} = A - \sigma\, I$. Oznaczmy przez $\overline{A}_k$ kolejne macierze powstające w tym procesie. Z wcześniejszej analizy metody QR wynika, że jeśli

$$|\lambda_{i^*} - \sigma| < \min_{i \neq i^*} |\lambda_i - \sigma|$$

to wyraz $\overline{a}^{(k)}_{n,n}$ w prawym dolnym rogu macierzy $\overline{A}_k$ zbiega do $\lambda_{i^*} - \sigma$ i zbieżność jest tym lepsza im σ jest bliższe λ_{i^*}. Dlatego ważne jest, aby σ wybrać tak, aby dobrze aproksymowała jedną z wartości własnych macierzy A. Oczywiście, przesunięcie możnaby w każdym kroku wybierać w zależności od aktualnie dostępnej informacji. Metoda wyglądałaby wtedy następująco:

$$\text{(QR-shift)} \qquad \begin{array}{l} A_0 := A; \quad Z_0 := I; \\ \text{dla} \quad k = 1, 2, 3, \ldots \\ \left\{ \begin{array}{l} A_{k-1} - \sigma_k\, I := Q_k\, R_k; \quad \text{(ortonormalizacja)} \\ A_k := R_k\, Q_k + \sigma_k\, I; \quad Z_k := Z_{k-1}\, Q_k; \end{array} \right. \end{array}$$

Jasne, że

$$A_k = Q_k^T\, (A_{k-1} - \sigma_k\, I)\, Q_k + \sigma_k\, I \;=\; Q_k^T\, A_{k-1}\, Q_k,$$

tzn. kolejne macierze A_k są ortogonalnie podobne. Okazuje się, że QR ma jeszcze jedną ważną własność. Oznaczmy przez $\vec{z}^{(k)}_n$ ostatnią kolumnę macierzy Z_k, przez $r^{(k)}_{n,n}$ wyraz w prawym dolnym rogu macierzy trójkątnej górnej R_k, oraz przez $\vec{e}_n$ ostatni wersor. Wtedy dla każdego k mamy

$$\begin{aligned} (A - \sigma_k\, I)\, \vec{z}^{(k)}_n &= Z_{k-1}\, (A_{k-1} - \sigma_{k-1}\, I)^T\, Z_{k-1}^T\, \vec{z}^{(k)}_n \\ &= Z_{k-1}\, (Q_k\, R_k)^T\, Z_{k-1}^T\, \vec{z}^{(k)}_n \\ &= Z_{k-1}\, R_k^T\, Z_k^T\, \vec{z}^{(k)}_n \;=\; Z_{k-1}\, R_k^T\, \vec{e}_n \\ &= r^{(k)}_{n,n}\, Z_{k-1}\, \vec{e}_n \;=\; r^{(k)}_{n,n}\, \vec{z}^{(k-1)}_n, \end{aligned}$$

czyli

$$\frac{z^{(k)}_n}{r^{(k)}_{n,n}} = (A - \sigma_k\, I)^{-1}\, \vec{z}^{(k-1)}_n.$$

To zaś oznacza, że w kolejnych krokach iteracji QR z przesunięciami σ_k ostatnia kolumna macierzy Z_k jest taka sama jak wektor $\vec{x}_k$ w odwrotnej metodzie potęgowej z wektorem startowym $\vec{e}_n$ i kolejnymi przesunięciami σ_k. A jeśli tak, to możemy wykorzystać wyniki rozdziału 2.4 i stwierdzić, że dobrym wyborem przesunięcia jest

$$\sigma_k = (\vec{z}^{(k)}_n)^T\, A\, \vec{z}^{(k)}_n = \vec{e}_n^T\, Z_k^T\, A\, Z_k\, \vec{e}_n = \vec{e}_n\, A_k\, \vec{e}_n = a^{(k)}_{n,n}.$$

Przy tak dobranym σ_k dostajemy asymptotycznie zbieżność sześcienną elementu $a^{(k)}_{n,n}$ do λ_{i^*}.

Dla informacji podamy jeszcze, że możliwy jest też wybór *przesunięcia Wilkinsona*, gdzie jako σ_k bierze się tą wartość własną macierzy 2×2,

$$\begin{bmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{n,n}^{(k)} \end{bmatrix},$$

która jest najbliższa $a_{n,n}^{(k)}$, a jeśli obie są jednakowo oddalone od $a_{n,n}^{(k)}$ to mniejszą z nich. Wtedy co prawda nie zwiększamy szybkości zbieżności, ale za to metoda jest zawsze zbieżna.

Pozostałe wartości własne macierzy A możemy obliczyć stosując *deflację*. Mianowicie, gdy już jesteśmy dostatecznie blisko wartości własnej λ_{i^*} (co zwykle osiąga się wykonując kilka kroków QR i sprawdza przez obliczenie residuum $\|A\,\vec{z}_n^{(k)} - a_{n,n}^{(k)}\,\vec{z}_n^{(k)}\|_2$) to uznajemy, że wyrazy w ostatniej kolumnie i ostatnim wierszu macierzy A_k, poza wyrazem $a_{n,n}^{(k)}$, są zerowe i redukujemy problem do macierzy formatu $(n-1)\times(n-1)$. Oczywiście, ten ogólny przepis wymaga właściwej implementacji.

4. Zagadnienie własne IV

W tym rozdziale zajmiemy się jeszcze innymi nich dotychczas poznanymi metodami znajdowania wartości i wektorów własnych. Ponieważ duże znaczenie mają w nich obroty płaszczyzn, zaczniemy właśnie od definicji tych przekształceń, ich własności i pierwszych zastosowań.

4.1. Obroty Givensa

4.1.1. Definicja obrotu

Macierz obrotu o kąt φ w płaszczyźnie rozpiętej na wersorach $\vec{e}_i, \vec{e}_j$, gdze $i < j$, jest postaci

$$O_{i,j} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos\varphi & & \sin\varphi & & \\ & & & \ddots & & & \\ & & -\sin\varphi & & \cos\varphi & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}.$$

Macierz ta różni się od jednostkowej jedynie wyrazami $o_{i,i} = c = o_{j,j}$ oraz $o_{i,j} = s = -o_{j,i}$. łatwo sprawdzić, że $O_{i,j}$ jest przekształceniem ortogonalnym, $O_{i,j}\, O_{i,j}^T = I$.

W obliczeniach numerycznych macierze obrotu służą najczęściej do wyzerowania określonej współrzędnej danego wektora. Rzeczywiście, obracając wektor $\vec{a} = [a_1, \ldots, a_n]^T$ w płaszczyźnie rozpiętej na $\vec{e}_i$ i $\vec{e}_j$ zmieniamy jedynie współrzędne i-tą i j-tą. Dokładniej, $O_{i,j}\, \vec{a} = \vec{b}$, gdzie

$$b_k = \begin{cases} a_i \cos\varphi + a_j \sin\varphi, & k = i, \\ -a_i \sin\varphi + a_j \cos\varphi, & k = j, \\ a_k, & k \neq i, j. \end{cases}$$

Stąd, przyjmując

$$\cos\varphi = \frac{a_i}{\sqrt{a_i^2 + a_j^2}}, \qquad \sin\varphi = \frac{a_j}{\sqrt{a_i^2 + a_j^2}},$$

albo liczby do nich przeciwne, dostajemy $b_i = \|\vec{a}\|_2^2$ albo $b_i = -\|\vec{a}\|_2^2$, oraz $b_j = 0$. Z powyższych wzorów wynika, że kąt obrotu φ można zawsze tak dobrać, aby $|\varphi| \leqslant \pi/2$.

Przekształcenia tak określone nazywamy *obrotami Givensa*.

Zauważmy, że stosując obroty Givensa kolejno w płaszczyznach rozpiętych na wektorach $\vec{e}_1$ i $\vec{e}_j$ dla $j = 2, 3, \ldots, n-1$ możemy przekształcić dany wektor $\vec{a}$ na kierunek pierwszego wersora, tzn.

$$O_{1,n} \cdots O_{1,3}\, O_{1,2}\, \vec{a} = \pm\|\vec{a}\|_2^2 \cdot \vec{e}_1.$$

Takie operacje wymagają wykonania $6n$ mnożeń/dodawań i $n-1$ pierwiastkowań. Przypomnijmy, że ten sam efekt można uzyskać kosztem około $2n$ mnożeń/dodawań i jednego pierwiastka

stosując odbicia Householdera. Wydaje się więc, że obroty Givensa opłaca się stosować jedynie wtedy, gdy wektor $\vec{a}$ ma niewiele (proporcjonalnie do n) współrzędnych niezerowych. Nie jest to jednak do końca prawdą. Istnieje bowiem pewna modyfikacja podanych wzorów autorstwa Gentlemana pozwalająca istotnie zmniejszyć koszt, którą teraz zaprezentujemy.

4.1.2. Algorytm Gentlemana

Przypuśćmy, że chcemy zastosować serię obrotów Givensa O_{i_k,j_k} dla $k = 1, 2, \ldots, m$, gdzie k-ty obrót jest wybrany tak, że j_k-ta współrzędna wektora

$$\vec{a}^{(k)} := O_{i_k,j_k} \cdots O_{i_1,j_1}\, \vec{a}$$

jest zerowa. Pomysł modyfikacji polega na przedstawieniu każdego z obrotów w postaci

$$O_{i_k,j_k} \;=\; D_k\, S_{i_k,j_k}\, D_{k-1}^{-1},$$

gdzie $D_0 = I$, a D_k są pewnymi nieosobliwymi macierzami diagonalnymi wybranymi tak, by mnożenie przez S_{i_k,j_k} było prostsze niż mnożenie przez O_{i_k,j_k}. Wtedy mamy

$$\begin{aligned} & O_{i_m,j_m} \cdots O_{i_1,j_1} \\ &= \; (D_m\, S_{i_m,j_m}\, D_{m-1}^{-1}) \cdots (D_2\, S_{i_2,j_2}\, D_1^{-1})\, (D_1\, S_{i_1,j_1}\, D_0^{-1}) \\ &= \; D_m\, S_{i_m,j_m} \cdots S_{i_2,j_2}\, S_{i_1,j_1}. \end{aligned}$$

Macierze diagonalne D_k, a tym samym i przekształcenia S_{i_k,j_k}, zdefiniujemy indukcyjnie dla $k = 1, 2, \ldots, m$. Załóżmy, że zdefiniowaliśmy już $D_{k-1} = \operatorname{diag}(d_1, \ldots, d_n)$. Niech O_{i_k,j_k} będzie obrotem o kąt φ oraz $c = \cos\varphi$, $s = \sin\varphi$.

Dla uproszczenia zapisu podstawimy $D := D_{k-1}$, $\hat{D} := D_k = \operatorname{diag}(\hat{d}, \ldots, \hat{d}_n)$ oraz $p := i_k$, $q := j_k$. Dalej mamy $O_{p,q} = O_{i_k,j_k} = (o_{i,j})_{i,j=1}^n$, $S_{p,q} = S_{i_k,j_k} = (s_{i,j})_{i,j=1}^n$. Ponieważ $S_{p,q} = \hat{D}^{-1}\, O_{p,q}\, D$ to

$$s_{i,j} \;=\; \frac{d_j}{\hat{d}_i}\, o_{i,j} \;=\; \begin{cases} cd_p/\hat{d}_p, & i = j = p, \\ cd_q/\hat{d}_q, & i = j = q, \\ sd_q/\hat{d}_p, & i = p, j = q, \\ -sd_p/\hat{d}_q, & i = q, j = p, \\ d_j/\hat{d}_i, & i = j \neq p, q, \\ 0, & \text{wpp.} \end{cases}$$

Przypomnijmy, że obrót O_{i_k,j_k} jest wybrany tak, że $a_q^{(k)} = 0$. Zauważmy, że dla

$$\vec{z} := D^{-1}\, \vec{a}^{(k-1)} = S_{i_{k-1},j_{k-1}} \cdots S_{i_1,j_1}\, \vec{a}$$

mamy

$$\vec{a}^{(k)} \;=\; O_{p,q}\, \vec{a}^{(k-1)} \;=\; (\hat{D}\, S_{p,q}\, D^{-1})\, (D\, \vec{z}) \;=\; \hat{D}\, S_{p,q}\, \vec{z}.$$

Stąd warunek $a_q^{(k)} = 0$ jest równoważny warunkowi $(S_{p,q}\, \vec{z})_q = 0$, czyli

$$s\, z_p d_p = c\, z_q d_q.$$

Jeśli teraz $z_q = 0$ to przyjmujemy $\hat{D} := D$ i $S_{p,q} := I$. W przeciwnym przypadku, uwzględniając równość $s^2 + c^2 = 1$, dostajemy

$$c^2 = 1/t, \quad t := 1 + \frac{d_q^2 z_q^2}{d_p^2 z_p^2}, \qquad\qquad s^2 = 1/t', \quad t' := 1 + \frac{d_p^2 z_p^2}{d_q^2 z_q^2}.$$

Mamy teraz dwa przypadki.
(i) Jeśli $0 < z_q^2 d_q^2 \leqslant z_p^2 d_p^2$ to kładziemy $\hat{d}_p = c d_p$, $\hat{d}_q = c d_q$ i $\hat{d}_l = d_l$ dla $l \neq p, q$. Wtedy $s_{l,l} = 1$ dla $l \neq p, q$ oraz

$$\begin{bmatrix} s_{p,p} & s_{p,q} \\ s_{q,p} & s_{q,q} \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ -\beta & 1 \end{bmatrix},$$

gdzie

$$\alpha = \frac{z_q d_q^2}{z_p d_p^2}, \qquad \beta = \frac{z_q}{z_p}.$$

(ii) Jeśli $0 \leqslant z_p^2 d_p^2 < z_q^2 d_q^2$ to kładziemy $\hat{d}_p = s d_q$, $\hat{d}_q = s d_p$ i $\hat{d}_l = d_l$ dla $l \neq p, q$. Wtedy $s_{l,l} = 1$ dla $l \neq p, q$ oraz

$$\begin{bmatrix} s_{p,p} & s_{p,q} \\ s_{q,p} & s_{q,q} \end{bmatrix} = \begin{bmatrix} \alpha' & 1 \\ -1 & \beta' \end{bmatrix},$$

gdzie

$$\alpha' = \frac{z_p d_p^2}{z_q d_q^2}, \qquad \beta' = \frac{z_p}{z_q}.$$

Macierz $S_{p,q}$ ma podobną strukturę jak $O_{p,q}$. Obecność dwóch jedynek pozwalaja jednak zaoszczędzić dwa mnożenia przy operacji $S_{p,q}\,\vec{x}$ w porównaniu do $O_{p,q}\,\vec{x}$. Ponadto, co jest ważniejsze, obliczanie wielkości α i β (ew. α' i β') definiujących $S_{p,q}$ nie wymaga pierwiastkowania. Rzeczywiście, w kolejnych krokach procesu obliczeniowego wystarczy pamiętać współczynniki $\hat{d}_l^2$. Współczynniki te wyliczane są rekurencyjnie zgodnie ze wzorami $\hat{d}_l^2 = d_l^2$ dla $l \neq p, q$, a dla $l = p, q$

$$\hat{d}_l^2 = \begin{cases} d_l^2/t & \text{jeśli (i)}, \\ d_l^2/t' & \text{jeśli (ii)}. \end{cases}$$

Zauważmy, jeszcze, że takie rozbicie na przypadki (i) i (ii) pozwala nie tylko uniknąć dzielenia przez zero, ale również uniknąć możliwego niedomiaru. Wielkość t jeśli zachodzi (i) oraz t' jeśli zachodzi (ii) jest w przedziale $[1/2, 1)$.

Możnaby się spytać gdzie jest zysk obliczeniowy. Na końcu procesu dysponujemy bowiem jedynie macierzą D_m^2 i aby pomnożyć przez D_m należy najpierw wykonać n pierwiastkowań. Nie jest to jednak zawsze prawdą.

Na przykład, jeśli chcemy przekształcić dany wektor $\vec{a}$ na kierunek wersora $\vec{e}_1$ przy pomocy obrotów Givensa to po wykonaniu serii $m = n - 1$ przekształceń

$$S_{1,n} \cdots S_{1,3}\, S_{1,2}\, \vec{a}$$

jedynie pierwsza współrzędna otrzymanego wektora nie jest zerowa. Mnożenie przez D_m sprowadza się więc do operacji na jego pierwszej współrzędnej, to zaś wymaga obliczenia tylko jednego pierwiastka kwadratowego.

Podobnie, używając obrotów Givensa można kosztem porównywalnym do algorytmu bazującego na odbiciach Householdera przekształcić daną nieosobliwą macierz kwadratową A do postaci trójkątnej górnej (albo, równoważnie, znaleźć jej rozkład ortogonalno-trójkątny). Rzeczywiście, dokonuje się tego wykonując najpierw na macierzy serię $m = (n-1)n/2$ przekształceń

$$S_{n-1,n}\,(S_{n-2,n}\,S_{n-2,n-1}) \cdots (S_{2,n} \cdots S_{2,3})\,(S_{1,n} \cdots S_{1,2})\,A,$$

gdzie $S_{i,j}$ jest tak dobrana, aby wyzerować współrzędną (j, i) aktualnej macierzy, a następnie mnożąc kolejne wiersze otrzymanej macierzy trójkątnej górnej przez pierwiastki z kolejnych elementów diagonalnych macierzy D_m^2. Całość wymaga więc obliczenia n pierwiastków kwadratowych, podobnie jak w algorytmie Householdera. (Należy przy tym pamiętać, aby nie wykonywać niepotrzebnie operacji na elementach zerowych!)

4.2. Metoda Jacobiego

Metoda Jacobiego jest najstarszą z istniejących metod numerycznych znajdowania wartości własnych macierzy symetrycznych. Ma jednak nie tylko znaczenie historyczne. Chociaż jest stosunkowo wolno zbieżna to można ją stosować gdy chcemy obliczyć wartości własne z bardzo dobrą dokładnością w obecności błędów zaokrągleń.

Zakładamy, że $A \in \mathbb{R}^{n,n}$ jest macierzą symetryczną. Podobnie jak w metodzie QR, metoda Jacobiego startuje z macierzy $A_0 = A$ i produkuje ciąg macierzy podobnych A_k dla $k = 1, 2, 3, \ldots$, który zbiega do macierzy diagonalnej

$$\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

składającej się z wartości własnych A. Różnica polega na tym, że pojedyncze kroki metody wykonuje się używając obrotów Givensa,

$$\begin{aligned} A_k &= O_{i_k,j_k}^T A_{k-1} O_{i_k,j_k} = O_{i_k,j_k}^T O_{i_{k-1},j_{k-1}}^T A_{k-2} O_{i_k,j_k} O_{i_{k-1},j_{k-1}} \\ &= \ldots = O_{i_k,j_k}^T \cdots O_{i_1,j_1}^T A_0 O_{i_1,j_1} \cdots O_{i_k,j_k}. \end{aligned}$$

Oznaczając $O_k = O_{i_1,j_1} \cdots O_{i_k,j_k}$ mamy więc $A_k = O_k^T A O_k$.

Jako miarę odchylenia od macierzy diagonalnej Λ w naturalny sposób przyjmiemy połowę sumy kwadratów elementów pozadiagonalnych, czyli

$$\omega_k := \omega(A_k) = \sum_{i<j} (a_{i,j}^{(k)})^2,$$

gdzie $(a_{i,j}^{(k)})_{i,j=1}^n$. Wielkość ω_k dobrze charakteryzuje również błąd przybliżenia wartości własnych macierzy A przez elementy diagonalne macierzy A_k. Rzeczywiście, niech

$$D_k = \operatorname{diag}(a_{1,1}^{(k)}, \ldots, a_{n,n}^{(k)})$$

i $R_k = A_k - D_k$. Wtedy

$$D_k = O_k^T A O_k - O_k^T O_k R_k O_k^T O_k = O_k^T (A + E_k) O_k,$$

gdzie $E_k = -O_k R_k O_k^T$ oraz

$$\|E_k\|_F = \|R_k\|_F = \sqrt{2\omega_k}.$$

Wyrazy diagonalne macierzy D_k są więc wartościami własnymi macierzy $A + E_k$, którą możemy potraktować jako zaburzoną macierz A. Korzystając z twierdzenia Weyla 1.3 dostajemy, że istnieje takie uporządkowanie $\lambda_{p(1)}, \ldots, \lambda_{p(n)}$ wartości własnych macierzy A, że dla wszystkich i mamy

$$|a_{i,i}^{(k)} - \lambda_{p(i)}| \leqslant \|E_k\|_2 \leqslant \|E_k\|_F = \sqrt{2\omega_k}.$$

Stąd wniosek, że poszczególne obroty O_{i_k,j_k} powinniśmy wybierać tak, aby maksymalizować różnicę $\omega_{k-1} - \omega_k$. Aby odpowiedzieć na pytanie który obrót osiąga ten cel, przeanalizujmy pojedynczy krok. Dla prostoty zapisu oznaczymy $p := i_k$, $q := j_k$, $A := A_{k-1}$, $\hat{A} := A_k = O_{p,q}^T A O_{p,q}$, gdzie $O_{p,q}$ jest obrotem o kąt φ, $c = \cos\varphi$, $s = \sin\varphi$, $c^2 + s^2 = 1$.

Zauważmy, że mnożenie z lewej strony przez obrót $O_{p,q}^T$ zmienia jedynie wiersze p-ty i q-ty macierzy, a mnożenie z prawej strony przez $O_{p,q}$ zmienia kolumny p-tą i q-tą. Dlatego dla $i \neq p, q$ i $j \neq p, q$ wyrazy $\hat{a}_{i,j}$ i $a_{i,j}$ są sobie równe. Jeśli zaś $i \neq p, q$ i $j = p, q$ to

$$\hat{a}_{i,p} = \hat{a}_{p,i} = c a_{i,p} - s a_{i,q}, \qquad \hat{a}_{i,q} = \hat{a}_{q,i} = s a_{i,p} + c a_{i,q},$$

a stąd

$$(\hat{a}_{i,p})^2 + (\hat{a}_{i,q})^2 = (a_{i,p})^2 + (a_{i,q})^2$$

i podobnie $(\hat{a}_{p,i})^2 + (\hat{a}_{q,i})^2 = (a_{p,i})^2 + (a_{q,i})^2$. Otrzymujemy

$$\omega_{k-1} - \omega_k = \omega(A) - \omega(\hat{A}) = (a_{p,q})^2 - (\hat{a}_{p,q})^2.$$

Dla danych $p < q$ wielkości c i s definiujące obrót należy więc wybrać tak, aby

$$\hat{a}_{p,q} = 0.$$

Jeśli $a_{p,q} = 0$ to powyższą równość realizuje obrót zerowy. Załóżmy więc, że $a_{p,q} \neq 0$. Bezpośredni rachunek pokazuje, że wtedy

$$\begin{bmatrix} \hat{a}_{p,p} & \hat{a}_{p,q} \\ \hat{a}_{q,p} & \hat{a}_{q,q} \end{bmatrix} = \begin{bmatrix} c^2 a_{p,p} - 2sca_{p,q} + s^2 a_{q,q} & (c^2 - s^2)a_{p,q} + sc(a_{p,p} - a_{q,q}) \\ (c^2 - s^2)a_{p,q} + sc(a_{p,p} - a_{q,q}) & s^2 a_{p,p} + 2sca_{p,q} + c^2 a_{q,q} \end{bmatrix}.$$

Stąd równość $\hat{a}_{p,q} = 0$ zachodzi wtedy gdy

$$(c^2 - s^2)a_{p,q} + sc(a_{p,p} - a_{q,q}) = 0.$$

Jeśli podstawimy $t = s/c = \operatorname{tg}\varphi$ to ostatnia równość jest równoważna równaniu kwadratowemu

$$t^2 + 2\,\beta\, t - 1 \;=\; 0, \qquad \beta = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}.$$

Rozwiązując otrzymujemy następujące wzory:

$$c = \frac{1}{\sqrt{1+t^2}}, \qquad s = t \cdot c, \qquad \text{gdzie} \quad t = \frac{\operatorname{sign}\beta}{|\beta| + \sqrt{1+\beta^2}}.$$

Zauważmy jeszcze, że taka postać rozwiązania pozwala obliczyć s i c z małym błędem względnym w arytmetyce maszyny cyfrowej.

Pozostaje do ustalenia jak wybierać kolejne (p, q). W oryginalnej wersji algorytmu (p, q) wybierane są tak, że

$$|a_{p,q}| = \max_{i<j} |a_{i,j}|.$$

Ponieważ wtedy $a_{p,q}^2 \geqslant \omega(A)/N$, gdzie $N = n(n-1)/2$ to

$$\omega(\hat{A}) = \omega(A) - a_{p,q}^2 \leqslant \omega(A)(1 - 1/N). \tag{4.1}$$

Mamy więc zapewnioną zbieżność liniową, chociaż iloraz zbieżności jest bliski jedynce dla dużych wymiarów n. Taki wybór jest jednak zbyt kosztowny, bo wymaga znalezienia w każdym kroku elementu największego co do modułu wśród N elementów. Dlatego stosowane są inne strategie. Na przykład,

$$(1,2), (1,3), (2,3), (1,4), (2,4), (3,4), \ldots, (1,i), \ldots, (i-1,i), \ldots, (1,n), \ldots, (n-1,n),$$

i dalej cyklicznie, aż do osiągnięcia żądanej dokładności. W tym przypadku kłopot polega na tym, że obroty wykonuje się również wtedy, gdy $a_{p,q}^2$ jest znacznie mniejsze od $\omega(A)$, co nie jest opłacalne. Aby temu zapobiec, można wprowadzić pewien próg δ i nie wykonywać obrotów gdy $a_{p,q}^2 < \delta$. Na przykład, możemy wziąć $\delta = \omega(A)/N$. Wtedy, po wykonaniu każdego kroku spełniona jest nierówność (4.1).

4.3. QR dla macierzy trójdiagonalnej

Metodę Jacobiego stosuje się do pełnej, rzeczywistej macierzy symetrycznej. W tym podrozdziale zatrzymamy się na przypadku, gdzie $A = A^T \in \mathbb{R}^{n,n}$ jest dodatkowo trójdiagonalna,

$$A = T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}.$$

Przypomnijmy, że każdą macierz symetryczną można sprowadzić do postaci trójdiagonalnej przez podobieństwa ortogonalne. W rozdziale 2.1.1 pokazaliśmy jak to zrobić przy pomocy odbić Householdera. Ale równie dobrze możemy użyć obrotów Givensa. Rzeczywiście, najpierw wybieramy obroty $O_{2,i}$ dla $i = 3, 4, \ldots, n$ tak, aby mnożenie

$$O_{2,n}^T \cdots O_{2,4}^T \, O_{2,3}^T \, A =: \hat{A}$$

spowodowało wyzerowanie kolejno elementów $(3,1), (4,1), \ldots, (n,1)$. (Pamiętamy o algorytmie Gentlemana!) Ponieważ mnożenie z prawej strony przez $O_{2,i}$ zmienia jedynie kolumny 2-gą oraz i-tą to po operacji

$$\hat{A} \, O_{2,3} \, O_{2,4} \cdots O_{2,n}$$

wyzerowane elementy w pierwszej kolumnie się nie zmienią. Dalej postępujemy indukcyjnie dla kolejnych kolumn, od 2-giej do $(n-2)$-giej, i ostatecznie dostajemy macierz trójdiagonalną $T = O^T \, A \, O$, gdzie

$$O \;=\; O_{2,3} \cdots O_{2,n} \, O_{3,4} \cdots O_{3,n} \cdots O_{n-2,n-1} \, O_{n-2,n} \, O_{n-1,n},$$

przy czym mnożenie z lewej przez $O_{i,j}^T$ zeruje element $(i-1, j)$.

Oczywiście, wszystkie metody dla macierzy pełnej można stosować również dla macierzy trójdiagonalnej. Jednak ze względu na prostszą strukturę macierzy niektóre metody znacznie się upraszczają. Pokażemy to na przykładzie metody QR z rozdziału 3.2.

W oryginalnej metodzie QR (z przesunięciami lub bez) rozkład ortogonalno-trójkątny macierzy $A = Q \, R$ dokonuje się przy pomocy odbić Householdera. Dla macierzy trójdiagonalnej T wygodnie jest użyć do tego celu obrotów Givensa $O_{i,i+1}^T$ zerujących kolejno elementy $(i+1, i)$ dla $i = 1, 2, \ldots, n-1$. Wtedy

$$R \;=\; (O_{n-1,n}^T \, O_{n-2,n-1}^T \cdots O_{2,3}^T \, O_{1,2}^T) \, T$$

jest macierzą trójkątną górną. Teraz istotne jest, że po wykonaniu drugiego kroku metody QR, czyli mnożenia

$$(O_{1,2} \, O_{2,3} \cdots O_{n-1,n}) \, R$$

otrzymana macierz $\hat{T}$ pozostaje trójdiagonalna. Rzeczywiście, skoro R jest trójkątna górna, a mnożenie z lewej strony przez $O_{i,i+1}$ kombinuje jedynie wiersze i-ty i $(i+1)$-szy to $\hat{T}$ jest macierzą górną Hessenberga, a ponieważ jest również symetryczna, to jest trójdiagonalna.

Koszt jednego kroku metody QR dla symetrycznej macierzy trójdiagonalnej T jest więc proporcjonalny do n.

Mamy własność ogólniejszą. W procesie iteracyjnym metody QR macierz trójdiagonalna pozostaje w każdym kroku trójdiagonalna niezależnie od zastosowanego algorytmu rozkładu na iloczyn ortogonalno-trójkątny. Wynika to w szczególności stąd, że, jak pamiętamy, rozkład

taki jest wyznaczony jednoznacznie z dokładnością do macierzy diagonalnej z elementami ± 1. (Jako ćwiczenie, można to pokazać bezpośrednio w przypadku gdy rozkład dokonywany jest przy użyciu odbić Householdera.)

Oczywiście wszystkie pozostałe fakty na temat metody QR pokazane wcześniej dla macierzy pełnych, w szczególności te dotyczące wyboru przesunięcia i szybkości zbieżności, pozostają w mocy gdy macierz jest trójdiagonalna.

4.4. Rozkład według wartości szczególnych (SVD)

Na końcu serii wykładów na temat zadania własnego zajmiemy się znajdowaniem wartości szczególnych danej macierzy, a to dlatego, że zadanie to i metody jego rozwiązania wiążą się ściśle z zadaniem i metodami znajdowania wartości własnych macierzy symetrycznych.

4.4.1. Twierdzenie o rozkładzie

Najpierw przypomnimy twierdzenie o rozkładzie według wartości szczególnych, czyli *SVD* (ang. *Singular Value Decomposition*). Będziemy zakładać, bez zmniejszenia ogólności, że liczba kolumn macierzy A nie przekracza liczby jej wierszy, co najczęściej występuje w praktyce. W przeciwnym przypadku wystarczy zastosować poniższe twierdzenie dla macierzy transponowanej A^T.

Twierdzenie 4.1. *Dla dowolnej macierzy $A \in \mathbb{R}^{m,n}$, gdzie $m \geqslant n$, istnieją macierze ortogonalne $U \in \mathbb{R}^{m,m}$ i $V \in \mathbb{R}^{n,n}$ takie, że*

$$A = U \Sigma V^T,$$

gdzie $\Sigma \in \mathbb{R}^{m,n}$ jest macierzą diagonalną,

$$\Sigma = \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix}, \qquad \hat{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathbb{R}^{n,n},$$

$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_k > \sigma_{k+1} = \cdots = \sigma_n = 0$, a k jest rzędem macierzy A.

Dowód. Ponieważ macierz $A^T A$ jest symetryczna i nieujemnie określona to istnieje baza ortonormalna $\{\vec{v}_i\}_{i=1}^n$ jej wektorów własnych, a odpowiadające jej wartości własne są nieujemne (patrz twierdzenia 1.1). Oznaczmy te wartości własne odpowiednio przez σ_i^2,

$$\sigma_1 \geqslant \cdots \geqslant \sigma_k > \sigma_{k+1} = \ldots = \sigma_n = 0,$$

przy czym $k = \operatorname{rank}(A^T A) = \operatorname{rank}(A)$. Zauważmy, że

$$(A\,\vec{v}_i)^T (A\,\vec{v}_j) = \vec{v}_i^T (A^T A)\,\vec{v}_j = \sigma_j^2 \cdot (\vec{v}_i^T\,\vec{v}_j) = \begin{cases} 0 & i \neq j, \\ \sigma_j^2 & i = j. \end{cases}$$

To oznacza, że wektory $\vec{u}_i := A\,\vec{v}_i/\sigma_i$, $1 \leqslant i \leqslant k$, tworzą układ ortonormalny i można go uzupełnić wektorami $\vec{u}_{k+1}, \ldots, \vec{u}_m$ do bazy ortonormalnej w $\mathbb{R}^m$.

Zdefiniujmy teraz macierze ortogonalne

$$V = [V_1, V_2] \in \mathbb{R}^{n,n}, \qquad V_1 = [\vec{v}_1, \ldots, \vec{v}_k] \in \mathbb{R}^{n,k}, \quad V_2 = [\vec{v}_{k+1}, \ldots, \vec{v}_n] \in \mathbb{R}^{n,n-k},$$

$$U = [U_1, U_2] \in \mathbb{R}^{m,m}, \qquad U_1 = [\vec{u}_1, \ldots, \vec{u}_k] \in \mathbb{R}^{m,k}, \quad U_2 = [\vec{u}_{k+1}, \ldots, \vec{u}_n] \in \mathbb{R}^{m,m-k}.$$

Wtedy, oznaczając $\Sigma' = \mathrm{diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k,k}$, mamy

$$U^T A V = \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} [A V_1, A V_2] = \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} [U_1 \Sigma', 0] = \begin{bmatrix} \Sigma' & 0 \\ 0 & 0 \end{bmatrix} = \Sigma,$$

albo, równoważnie, $A = U \Sigma V^T$. □

Wielkości σ_i, $1 \leqslant i \leqslant n$, to właśnie *wartości szczególne* macierzy A. Są one wyznaczone jednoznacznie. Rzeczywiście, jeśli mamy dwa rozkłady, $U_1 \Sigma_1 V_1^T = A = U_2 \Sigma_2 V_2^T$, to

$$V (\Sigma_1^T \Sigma_1) = (\Sigma_2^T \Sigma_2) V, \qquad V = V_2^T V_1.$$

Porównując wyrazy diagonalne po obu stronach powyższej równości dostajemy $\sigma_i^{(1)} = \sigma_i^{(2)}$ dla wszsytkich i.

Zanotujmy jeszcze, że mając rozkład SVD można np. łatwo rozwiązać liniowe zadanie najmniejszych kwadratów, tzn. znaleźć wektor $\vec{x}^*$ minimalizujący $\|\vec{b} - A\vec{x}\|_2$ po wszystkich $\vec{x} \in \mathbb{R}^n$. (Jeśli istnieje wiele takich wektorów to bierzemy ten o najmniejszej normie.) Mamy bowiem

$$\|\vec{b} - A\vec{x}\|_2 = \|U U^T \vec{b} - U \Sigma V^T \vec{x}\|_2 = \|\vec{c} - \Sigma \vec{y}\|_2,$$

$\vec{c} = U^T \vec{b}$, $\vec{y} = V^T \vec{x}$. Stąd $\vec{x}^* = V^T \vec{y}^*$, gdzie

$$y_i^* = \begin{cases} c_i/\sigma_i, & 1 \leqslant i \leqslant k, \\ 0, & 1 \leqslant i \leqslant n. \end{cases}$$

4.4.2. Dlaczego nie pomnożyć $A^T A$?

Z dowodu twierdzenia 4.1 wynika, że $\sigma_1^2, \ldots, \sigma_n^2$ są wartościami własnymi macierzy symetrycznej i nieujemnie określonej $A^T A$, a kolumny macierzy ortogonalnej V tworzą odpowiednio bazę ortonormalną w $\mathbb{R}^n$ wektorów własnych tej macierzy. Podobnie, $\sigma_1^2, \ldots, \sigma_n^2, \underbrace{0, \ldots, 0}_{m-n}$ są wartościami własnymi $A A^T$, a kolumny U tworzą bazę ortonormalną w $\mathbb{R}^m$ wektorów własnych. Wydaje się więc, że wartości szczególne (i w razie potrzeby cały rozkład SVD) można łatwo wyznaczyć wykonując najpierw mnożenie $\hat{A} := A^T A$ lub $\hat{A} := A A^T$, a następnie aplikując do macierzy $\hat{A}$ jedną z rozpatrzonych wcześniej metod dla zadania własnego. Należy jednak przestrzec przed takim mechanicznym działaniem.

Przykład 4.1. Niech $A = \begin{bmatrix} 1 & 1 \\ 0 & \varepsilon \end{bmatrix}$. Dla „małych” ε wartości szczególne tej macierzy są bliskie $\sqrt{2}$ i $|\varepsilon|/\sqrt{2}$. Jeśli ε jest na tyle małe, że $1 + \varepsilon^2$ jest w arytmetyce *fl* reprezentowane przez 1 to macierz $A^T A = \begin{bmatrix} 1 & 1 \\ 1 & 1 + \varepsilon^2 \end{bmatrix}$ jest reprezentowana przez macierz $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, która jest już osobliwa - jej druga wartość szczególna jest zerowa.

Z uwagi na możliwą zmianę rzędu macierzy, jak w przytoczonym przykładzie, stosuje się nieco zmodyfikowane algorytmy znajdowania wartości własnych, które unikają jawnych mnożeń $A^T A$ lub $A A^T$. Pokażemy to najpierw, nie wdając się w szczegóły, na przykładzie metody Jacobiego z rozdziału 4.2.

Przypomnijmy, że metoda Jacobiego zastosowana bezpośrednio do macierzy $\hat{A} = A^T A$ konstruuje ciąg macierzy podobnych $\hat{A} = \hat{A}_0, \hat{A}_1, \ldots, \hat{A}_k, \ldots$, gdzie $\hat{A}_k = O_{i_k,j_k}^T \hat{A}_{k-1} O_{i_k,j_k}$, a O_{i_k,j_k}

jest tak dobranym obrotem Givensa, aby wyzerować element (i_k, j_k) i jednocześnie, wobec symetrii, także (j_k, i_k). Oznaczmy $A_0 = A$ i $A_k = A_{k-1}\, O_{i_k,j_k}$ dla $k \geqslant 1$. Wtedy $\hat{A}_k = A_k^T\, A_k$. Pomysł polega na tym, aby zamiast obliczać jawnie $\hat{A}_k = O_{i_k,j_k}^T\, A_{k-1}^T\, A_{k-1}\, O_{i_k,j_k}$, w kolejnych krokach obliczać i pamiętać jedynie $A_k = A_{k-1}\, O_{i_k,j_k}$. Jest to możliwe, bowiem wyznaczenie rotacji O_{i_k,j_k} wymaga jedynie znajomości elementów

$$\hat{a}_{i_k,i_k}^{(k-1)} = \sum_{l=1}^{m} (a_{l,i_k}^{(k-1)})^2, \qquad \hat{a}_{j_k,j_k}^{(k-1)} = \sum_{l=1}^{m} (a_{l,j_k}^{(k-1)})^2, \qquad \hat{a}_{i_k,j_k}^{(k-1)} = \sum_{l=1}^{m} a_{l,i_k}^{(k-1)} a_{l,j_k}^{(k-1)},$$

które można obliczyć korzystając z powyższych wzorów.

4.4.3. SVD dla macierzy dwudiagonalnych

W tym podrozdziale pokażemy algorytm, który można traktować jako wariant metody QR zastosowanej do macierzy $A^T\, A$. Zanim jednak przejdziemy do samego algorytmu, poczynimy kilka pomocniczych uwag teoretycznych.

Niech T_0 będzie macierzą symetryczną i dodanio okteśloną, $T_0 = T_0^T > 0$. Rozpatrzmy proces iteracyjny, nazwijmy go LR, który startuje z macierzy T_0 i w kolejnych krokach $k = 1, 2, \ldots$
(i) wybiera przesunięcie τ_k, mniejsze od najmniejszej wartości własnej T_{k-1} (np. $\tau_k = 0$),
(ii) dokonuje rozkładu Banachiewicza-Cholesky'ego

$$T_{k-1} - \tau_k^2 \cdot I = B_k^T\, B_k,$$

gdzie B_k jest macierzą trójkątną górną z dodatnimi elementami na głównej przekątnej,
(iii) produkuje $T_k = B_k\, B_k^T + \tau_k^2 \cdot I$.
(Przypomnijmy, że rozkładu Banachiewicza-Cholesky'ego dokonujemy zmodyfikowanym algorytmem eliminacji Gaussa.) Oczywiście, macierze T_k są do siebie podobne, bo

$$T_k = B_k\, B_k^T + \tau_k^2 \cdot I = B_k\, (T_{k-1} - \tau_k^2 \cdot I)\, B_k^{-1} + \tau_k^2 \cdot I = B_k\, T_{k-1}\, B_k^{-1}.$$

Zachodzi ciekawsza własność.

Lemat 4.1. *Dwa kroki iteracji LR z tym samym przesunięciem τ są równoważne jednemu krokowi iteracji QR z przesunięciem τ.*

Dowód. Bez zmniejszenia ogólności przyjmijmy, że $k = 0$ i $\tau_1 = \tau_2 = \tau$. Z jednej strony, z rozkładu QR mamy $(T_0 - \tau^2 \cdot I)^2 = (Q\, R)^T\, (Q\, R) = R^T\, R$, gdzie elementy na głównej przekątnej macierzy R są dodatnie, a z drugiej z rozkładu LR

$$\begin{aligned} (T_0 - \tau^2 \cdot I)^2 &= (B_1^T\, B_1)\, (B_1^T\, B_1) \,=\, B_1^T\, (T_1 - \tau^2 \cdot I)\, B_1 \\ &= B_1^T\, B_2^T\, B_2\, B_1 \,=\, (B_2\, B_1)^T\, (B_2\, B_1). \end{aligned}$$

Wobec jednoznaczności rozkładu Banachiewicza-Cholesky'ego mamy więc $R = B_2\, B_1$. Stąd macierz powstała w wyniku jednego kroku iteracji QR wynosi

$$\begin{aligned} \hat{T} &= R\, Q + \tau^2 \cdot I \,=\, R\, (Q\, R)\, R^{-1} + \tau^2 \cdot I \\ &= R\, (T_0 - \tau^2 \cdot I)\, R^{-1} + \tau^2 \cdot I \,=\, R\, T_0\, R^{-1} \\ &= (B_2\, B_1)\, (B_1^T\, B_1 + \tau^2 \cdot I)\, (B_2\, B_1)^{-1} \\ &= B_2\, B_1\, B_1^T\, B_1\, B_1^{-1}\, B_2^{-1} + \tau^2 (B_2\, B_1)\, (B_2\, B_1)^{-1} \\ &= B_2\, (B_1\, B_1^T)\, B_2^{-1} + \tau^2 \cdot I \,=\, B_2\, (T_1 - \tau^2 \cdot I)\, B_2^{-1} + \tau^2 \cdot I \\ &= B_2\, (B_2^T\, B_2)\, B_2^{-1} + \tau^2 \cdot I \,=\, B_2\, B_2^T + \tau^2 \cdot I \\ &= (T_2 - \tau^2 \cdot I) + \tau^2 \cdot I \,=\, T_2, \end{aligned}$$

jak w tezie lematu. □

Powyższy lemat pokazuje, że rozważania teoretyczne dotyczące iteracji QR można przenieść na iteracje LR. Dlatego dalej nie będziemy się już zajmować analizą teoretyczną LR, a przejdziemy do opisu algorytmu te iteracje wykorzystującego.

Zakładamy, że $A \in \mathbb{R}^{m,n}$ jest kolumnowo regularna, czyli $\operatorname{rank}(A) = n$, oraz dwudiagonalna. Dokładniej, $a_{i,j} = 0$ dla $i \geqslant j+1$ i dla $j \geqslant i+2$,

$$A = \begin{bmatrix} a_1 & b_1 & & \\ & a_2 & \ddots & \\ & & \ddots & b_{n-1} \\ & & & a_n \\ & & & \end{bmatrix}.$$

Oczywiście, wobec kolumnowej regularności macierzy mamy $a_j \neq 0$. Założenie o dwudiagonalności wydaje się mocno ograniczające. W istocie jednak tak nie jest, bo każdą macierz można sprowadzić do takiej postaci nie zmieniając jej wartości szczególnych i kontrolując odpowiednie wektory własne, co pokażemy na samym końcu.

Algorytm działa podobnie jak iteracje LR z przesunięciami, ale zamiast tworzyć jawnie macierze T_k, podobne do $A^T A$, pracuje bezpośrednio na macierzach B_k. Przyjmujemy $B_0 = D\,A$, gdzie $D \in \mathbb{R}^{n,m}$ jest macierzą diagonalną z elementami na głównej przekątnej $d_j = \operatorname{sign}(a_j)$, $1 \leqslant j \leqslant n$. Wtedy $B_0 \in \mathbb{R}^{n,n}$ jest macierzą dwudiagonalną z dodatnimi elementami na głównej przekątnej oraz

$$T_0 = B_0^T\, B_0 = A^T\,(D^T\, D)\, A = A^T\, A$$

jest macierzą trójdiagonalną. Zobaczmy teraz jak mając macierz dwudiagonalną B_k możemy skonstruować B_{k+1}. Ponieważ T_k jest trójdiagonalna to B_{k+1} musi być dwudiagonalna. (W szczególności, wyniknie to również z dalszych rachunków.) Dla uproszczenia zapisu, oznaczmy elementy przekątniowe macierzy B_k i B_{k+1} odpowiednio przez a_j i $\hat{a}_j$, $1 \leqslant j \leqslant n$, a elementy nad główną przekątną przez b_j i $\hat{b}_j$, $1 \leqslant j \leqslant n-1$. Przyjmijmy dodatkowo $b_0 = \hat{b}_0 = b_n = \hat{b}_n = 0$. Macierze B_k i B_{k+1} związane są równaniem

$$B_{k+1}^T\, B_{k+1} + \tau_{k+1}^2 \cdot I \;=\; T_k \;=\; B_k\, B_k^T + \tau_k^2 \cdot I.$$

Porównując wyrazy przekątniowe po obu stronach tej równości otrzymujemy

$$\hat{a}_j^2 + \hat{b}_{j-1}^2 + \tau_{k+1}^2 = a_j^2 + b_j^2 + \tau_k^2, \qquad 1 \leqslant j \leqslant n,$$

a porównując wyrazy nad przekątną

$$\hat{a}_j \hat{b}_j = a_{j+1} b_j^2, \qquad 1 \leqslant j \leqslant n-1.$$

Stąd, podstawiając $\delta = \tau_{k+1}^2 - \tau_k^2$, mamy następujące wzory na $\hat{a}_j$ i $\hat{b}_j$. Dla $j = 1, 2, \ldots, n-1$ obliczamy

$$\hat{a}_j := \sqrt{a_j^2 + b_j^2 - \hat{b}_{j-1}^2 - \delta}, \qquad \hat{b}_j := a_{j+1} b_j / \hat{a}_j$$

i na końcu $\hat{a}_n := \sqrt{a_n^2 - \hat{b}_{n-1}^2 - \delta}$.

Ponieważ obliczanie pierwiastków jest stosunkowo kosztowne, wzory te opłaca się zmodyfikować tak, aby nie obliczać pierwiastków w każdej iteracji, a jedynie na końcu całego procesu iteracyjnego. Można to osiągnąć pracując na kwadratach a_j i b_j. Rzeczywiście, wprowadzając zmienne $p_j = a_j^2$ i $q_j = b_j^2$, otrzymujemy następującą procedurę dla jednego kroku iteracyjnego:

```
begin
    for j = 1 to n − 1 do
    begin
        p̂_j := p_j + q_j − q̂_j − δ;
        q̂_j := q_j · (q_{j+1}/q̂_j)
    end;
    p̂_n := p_n − q̂_{n−1} − δ
end;
```

Koszt jednego kroku iteracyjnego jest stały. Stąd, jeśli macierz wyjściowa A jest dwudiagonalna to całkowity koszt algorytmu jest proporcjonalny do liczby iteracji.

A co jeśli A nie jest dwudiagonalna? Wtedy trzeba ją wstępnie przekształcić do postaci dwudiagonalnej nie zmieniając wartości szczególnych. W tym celu możemy użyć np. odbić Householdera. Najpierw zerujemy wyrazy w pierwszej kolumnie, oprócz wyrazu diagonalnego, poprzez pomnożenie macierzy A z lewej strony przez odbicie $H_{L,1} \in \mathbb{R}^{m,m}$ przekształcające pierwszą kolumnę A na kierunek $\vec{e}_1$. Oznaczmy $\tilde{A}_1 = (\tilde{a}_{i,j}) = H_{L,1}\, A$. Następnie wybieramy odbicie $\tilde{H}_{R,1} \in \mathbb{R}^{n-1,n-1}$ tak, aby przekształcało wektor $[\tilde{a}_{1,2}, \tilde{a}_{1,3} \ldots, \tilde{a}_{1,n}]^T \in \mathbb{R}^{n-1}$ na kierunek $\vec{e}_1 \in \mathbb{R}^{n-1}$ i mnożymy $\tilde{A}_1$ z prawej przez $H_{R,1}^T = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{0} & \tilde{H}_{R,1}^T \end{bmatrix} \in \mathbb{R}^{n,n}$. Ponieważ to mnożenie nie zmienia pierwszej kolumny $\tilde{A}_1$, w powstałej macierzy $A_1 = H_{L,1}\, A\, H_{R,1}^T$ elementy w pierwszej kolumnie i w pierwszym wierszu, poza $(1,1)$ i $(1,2)$, są wyzerowane. Dalej postępujemy indukcyjnie zerując na przemian odpowiednie elementy w kolumnach i wierszach. Ostatecznie, po $n-1$ krokach otrzymujemy macierz

$$\hat{A} = U_1\, A\, V_1^T, \quad \text{gdzie} \quad U_1 = H_{L,n-1} \cdots H_{L,1}, \quad V_1 = H_{R,n-2} \cdots H_{R,1},$$

która jest już dwudiagonalna. Jeśli teraz $\hat{A} = U_2\, \Sigma\, V_2^T$ to $A = (U_1^T\, U_2)\, \Sigma\, (V_1^T\, V_2)^T$, a więc A i $\hat{A}$ mają te same wartości szczególne. Ponadto,

$$A = U\, \Sigma\, V^T, \quad \text{gdzie} \quad U = U_1^T\, U_2, \quad V = V_1^T\, V_2.$$

Zanotujmy jeszcze, że dowolną macierz można sprowadzić w podobny sposób do postaci dwudiagonalnej przy pomocy obrotów Givensa. Zarówno w przypadku zastosowania obrotów Givensa jak i odbić Householdera koszt jest proporcjonalny do $m \cdot n^2$.

5. Proste metody iteracyjne rozwiązywania układów równań liniowych

Zajmiemy się przybliżonymi metodami rozwiązywania układów równań liniowych

$$Ax = b$$

z nieosobliwą rzeczywistą macierzą A. Jak już sugerowaliśmy we wstępie, wraz z coraz większymi modelami pojawiającymi się w praktyce obliczeniowej, częściej zachodzi potrzeba rozwiązywania zadań algebry liniowej o bardzo wielkiej liczbie niewiadomych. Gdy A jest dużego rozmiaru, metody bezpośrednie (wykorzystujące na przykład rozkład LU metodą eliminacji Gaussa lub inne rozkłady macierzy, np. QR) mogą być zbyt kosztowne lub mniej wygodne w użyciu. Przykładowo, jeśli niewiadomych jest kilka milionów, a macierz nie ma żadnej specjalnej struktury, to korzystając z eliminacji Gaussa musielibyśmy wykonać rzędu 10^{19} działań arytmetycznych (i zarezerwować rzędu 10^{13} bajtów — czyli 10000 GB — pamięci na czynniki rozkładu). Zakładając optymistycznie, że nasz komputer jest w stanie wykonać 10^{10} działań na sekundę (czyli jest w stanie realnie osiągnąć wydajność 10 Gflopów/s), dawałoby to 10^9 sekund, czyli około 32 lat... Rozsądną alternatywą może być wtedy rozwiązanie układu w sposób *przybliżony*, ale za to znacznie tańszy lub wygodniejszy. Jednym ze sposobów przybliżonego rozwiązywania układów równań liniowych są metody iteracyjne, szczególnie użyteczne wówczas, gdy A jest macierzą rozrzedzoną.

5.1. Macierze rozrzedzone

Macierze, w których bardzo wiele elementów jest zerowych, nazywamy *macierzami rozrzedzonymi* lub, potocznie, *rzadkimi*. Dla odróżnienia, macierze, które nie są rzadkie, nazwiemy *gęstymi* — przykładem takiej macierzy jest macierz, której wszystkie elementy są niezerowe. Intuicyjnie możemy spodziewać się, że praktyczne zadania liniowe *wielkiego wymiaru* będą prowadziły właśnie do macierzy rozrzedzonej, gdyż związki pomiędzy niewiadomymi w pojedynczym równaniu zadanego układu nie będą raczej dotyczyć wszystkich, tylko wybranej (nielicznej) grupy innych niewiadomych.

Wykorzystanie rozrzedzenia macierzy nie tylko może doprowadzić do algorytmów istotnie szybszych od ich analogonów dla macierzy gęstych (jak pamiętamy, standardowe algorytmy oparte na rozkładzie macierzy gęstej, np. LU, potrzebują $O(N^3)$ działań arytmetycznych), ale wręcz może być jedynym realnym sposobem na to, by niektóre zadania w ogóle stały się *rozwiązywalne* przy obecnym stanie techniki obliczeniowej!

Gdy A jest rozrzedzona, mnożenie takiej macierzy przez wektor jest bardzo tanie (koszt jest proporcjonalny do liczby niezerowych elementów macierzy). Dlatego, konstruując metodę *przybliżonego* rozwiązywania układu, warto oprzeć się na iteracji, których głównym składnikiem jest operacja mnożenia przez A lub jej część.

5.1.1. Przykłady macierzy rozrzedzonych

Jak zobaczymy poniżej, rzeczywiście łatwo można spotkać realne zadania matematyki stosowanej, w których macierz wymiaru N ma tylko $O(N)$ niezerowych elementów. Omówimy tu dwie klasy takich zadań: dyskretyzacje równań różniczkowych oraz łańcuchy Markowa z dyskretną przestrzenią stanów.

Równania różniczkowe cząstkowe Jednym ze szczególnie ważnych źródeł układów równań z macierzami rozrzedzonymi są *równania różniczkowe cząstkowe* (pochodzące np. z modeli pogody, naprężeń w konstrukcji samochodu, przenikania kosmetyków do głębszych warstw skóry, itp.).

Przykład 5.1 (Dyskretyzacja jednowymiarowego laplasjanu)**.** Rozważmy modelowe eliptyczne równanie różniczkowe z jednorodnym warunkiem brzegowym Dirichleta

$$-u''(x) = f(x) \qquad \forall x \in (0,1), \tag{5.1}$$
$$u(0) = u(1) = 0, \tag{5.2}$$

w którym u jest szukanym rozwiązaniem, a f — zadaną funkcją. Aby znaleźć jego przybliżone rozwiązanie, możemy na przykład zastąpić równanie różniczkowe odpowiadającym mu równaniem *różnicowym*,

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(x_i) \qquad \forall i = 1, \ldots, N, \tag{5.3}$$
$$u_0 = u_{N+1} = 0, \tag{5.4}$$

gdzie u_i ma odpowiadać wartości rozwiązania w węźle dyskretyzacji x_i, $u_i \approx u(x_i)$, oraz $x_i = i \cdot h$, przy czym $h = 1/(N+1)$. Odpowiedzią na pytanie, jak dobrze (i czy w ogóle) takie rozwiązanie *dyskretne* aproksymuje rozwiązanie dokładne, zajmujemy się na wykładzie z Numerycznych równań różniczkowych; tutaj zobaczymy, do jakiego zagadnienia algebraicznego prowadzi równanie różnicowe (5.3).

Jest to oczywiście równanie liniowe na współczynniki $U_N = (u_1, \ldots, u_N)^T$,

$$T_N U_N = h^2 F_N,$$

gdzie

$$T_N = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}_{N \times N} \tag{5.5}$$

oraz $F_N = (f(x_1), \ldots, f(x_N))^T$. Ponieważ interesują nas dobre przybliżenia (to znaczy przypadek, gdy h jest *bardzo* małe), znaczy to, że N będzie bardzo duże. Nasza macierz jest rzeczywiście macierzą rozrzedzoną, bo ma nieco mniej niż $3N$ niezerowych elementów. Jej współczynnik wypełnienia (ang. *density*), czyli stosunek liczby niezerowych do liczby wszystkich elementów macierzy, jest rzędu $3/N$ i maleje ze wzrostem N.

Otrzymana powyżej macierz, aczkolwiek ma bardzo wiele cech charakterystycznych dla dyskretyzacji „poważniejszych" równań różniczkowych, wydaje się jednak zbyt trywialna, by traktować ją metodą inną niż eliminacja Gaussa (uwzględniając jej pasmową strukturę): rzeczywiście, jest to macierz trójdiagonalna (a przy tym: symetryczna i dodatnio określona), zatem właściwie zrealizowany algorytm eliminacji Gaussa (lub lepiej: rozkładu LDL^T) pozwoli nam

wyznaczyć rozwiązanie układu $T_N U_N = h^2 F_N$ kosztem $O(N)$, a więc — z dokładnością do stałej — optymalnym.

Jednak, gdy przejdziemy do wyższych wymiarów i być może dodatkowo regularną siatkę węzłów dyskretyzacji zastąpimy na przykład nieregularną siatką elementu skończonego, uzyskamy macierze o znacznie bardziej skomplikowanej strukturze, które już nie tak łatwo poddają się eliminacji Gaussa.

Przykład 5.2 (Dyskretyzacja wielowymiarowego laplasjanu)**.** Przez analogię z poprzednim przykładem, rozważmy równanie Poissona w obszarze $\Omega \subset \mathbb{R}^d$, z jednorodnym warunkiem Dirichleta:

$$-\Delta u(x) = f(x) \qquad \forall x \in \Omega, \tag{5.6}$$

$$u_{|\partial\Omega} = 0, \tag{5.7}$$

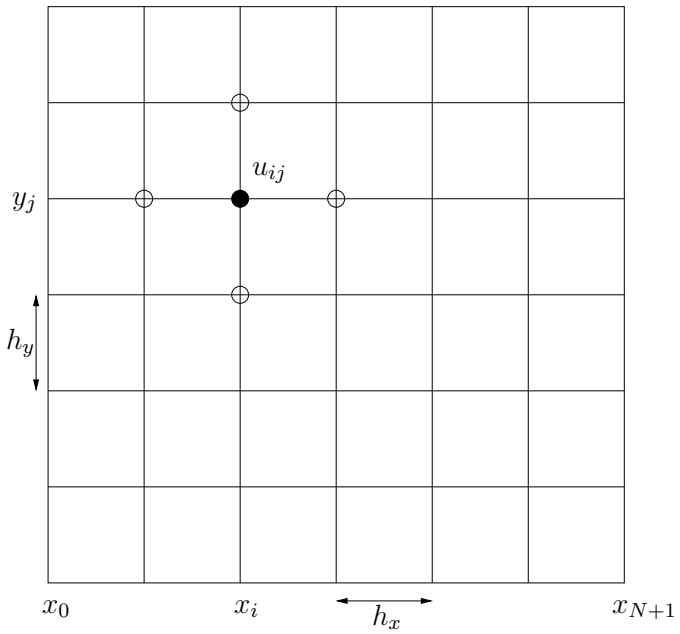
w którym $u : \Omega \to \mathbb{R}$ jest szukanym rozwiązaniem, a $f : \Omega \to \mathbb{R}$ — zadaną funkcją. Δ oznacza operator Laplace'a,

$$\Delta \equiv \frac{\partial^2}{\partial x_1^2} + \ldots + \frac{\partial^2}{\partial x_d^2}.$$

W dalszym ciągu przyjmiemy, dla ominięcia dodatkowych trudności, że Ω jest kostką jednostkową: $\Omega = (0,1)^d$.

Aby znaleźć przybliżone rozwiązanie, znów możemy na przykład zastąpić równanie różniczkowe odpowiadającym mu równaniem *różnicowym*. Wybierzemy tu najmniej efektywną, ale za to koncepcyjnie najprostszą metodę dyskretyzacji, opartą na równomiernej siatce węzłów w Ω.

Przykładowo, jeśli $d = 2$, to weźmiemy siatkę węzłów $p_{ij} = (x_i, y_j) \in \Omega$, gdzie $x_i = ih_x, y_j = jh_y$, przy czym $h_x = a/(n_x + 1)$ jest krokiem siatki w kierunku x i analogicznie $h_y = b/(n_y + 1)$, zob. rysunek 5.1.



Rysunek 5.1. Regularna siatka na Ω.

Zastępując pochodne ilorazami różnicowymi [8]

$$\Delta u(x_i, y_j) \approx \Delta_h u_{ij} = \frac{1}{h^2}(4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}), \tag{5.8}$$

dostajemy równanie różnicowe

$$-\frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{h_x^2}-\frac{u_{i,j-1}-2u_{i,j}+u_{i,j+1}}{h_y^2}=f_{ij}\qquad \forall i=1,\ldots,n_x,\ j=1,\ldots,n_y. \tag{5.9}$$

W pozostałych węzłach siatki rozwiązanie przyjmuje wartość równą zero na mocy warunku brzegowego Dirichleta. Więcej na temat metod dyskretyzacji takich i podobnych równań różniczkowych można dowiedzieć się na wykładzie Numeryczne równania różniczkowe.

W dalszym ciągu, dla uproszczenia zapisu będziemy zakładać, że $n_x = n_y = n$ i w konsekwencji $h_x = h_y = h$. W takim układzie mamy do wyznaczenia $N = n^2$ niewiadomych:

$$U_N = (\underbrace{u_{1,1}, u_{2,1}, u_{3,1}, \ldots, u_{N_x,1}}_{\text{niewiadome pierwszej kolumny}}, \underbrace{u_{1,2}, u_{2,2}, u_{3,2}, \ldots, u_{N_x,2}}_{\text{drugiej kolumny}}, \ldots, \underbrace{u_{1,N_y}, u_{2,N_y}, u_{3,N_y}, \ldots, u_{N_x,N_y}}_{\text{ostatniej kolumny}})^T.$$

które spełniają liniowy układ równań o postaci macierzowej

$$P_N\, U_N = h^2 F_N,$$

gdzie P_N jest tym razem pięciodiagonalną macierzą o blokowej strukturze,

$$P_N = \begin{pmatrix} T_n+2I & -I & & & \\ -I & T_n+2I & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T_n+2I & -I \\ & & & -I & T_n+2I \end{pmatrix}_{N\times N}. \tag{5.10}$$

W kolejnych diagonalnych blokach P_N znajdują się macierze trójdiagonalne T_n+2I, gdzie T_n jest znaną już nam macierzą dyskretyzacji jednowymiarowego laplasjanu. Używając więc symbolu Kroneckera dla iloczynu tensorowego macierzy mamy przy naszej numeracji niewiadomych

$$P_N = I \otimes T_n + T_n \otimes I.$$

W przypadku tej macierzy, prosta strategia polegająca na eliminacji Gaussa (z wykorzystaniem dodatkowo faktu, że P_N jest pasmowa, o szerokości pasma $2n$), dałaby nam szansę rozwiązać to równanie kosztem $O(n^4)$, co jest, przy dużych wielkościach n, wartością trudną do zaakceptowania.

Prowadząc analogicznie dyskretyzację na jednorodnej siatce *trójwymiarowego* zadania Poissona na kostce jednostkowej ($d = 3$), dostalibyśmy $N = n^3$ oraz zadanie z macierzą siedmiodiagonalną (o szerokości pasma $2n^2$):

$$S_N = I \otimes I \otimes T_n + I \otimes T_n \otimes I + T_n \otimes I \otimes I. \tag{5.11}$$

Nawet dla średnich wartości n, układ ten trudno rozwiązać metodą bezpośrednią, opartą na prostym rozkładzie pasmowej macierzy (tutaj na przykład: rozkładzie Cholesky'ego), zarówno ze względu na koszt obliczeniowy, jak i koszt pamięciowy. Rzeczywiście, koszt rozkładu Cholesky'ego uwzględniającego to, że w czynnikach rozkładu poza pasmem nie pojawią się elementy niezerowe wyniesie $O(n^4)$ w przypadku dwuwymiarowego równania Poissona i $O(n^7)$ w przypadku trójwymiarowego równania Poissona (dlaczego?). To oczywiście mniej niż pesymistyczne $O(N^3) = O(n^9)$, ale gdy n jest duże, może być wystarczająco zachęcające do tego, by poszukać tańszych rozwiązań, dających sensowne rozwiązanie w sensownym czasie.

Przykład 5.3 (ewolucyjne równanie dyfuzji)**.** Rozważmy równanie

$$\begin{aligned} u_t(x,t) - \Delta_x u(x,t) &= f(x,t) && \text{dla } (x,t) \in \Omega \times (0,T), \\ u(x,t) &= 0 && \text{dla } (x,t) \in \partial\Omega \times [0,T). \end{aligned}$$

Dla uproszczenia przyjmiemy, że Ω jest kwadratem jednostkowym. Wprowadzając dyskretyzację tego równania po zmiennej przestrzennej jak poprzednio, a po zmiennej czasowej — niejawny schemat Eulera (patrz rozdział o schematach różnicowych Numerycznych równań różniczkowych), otrzymujemy, po uwzględnieniu warunku brzegowego, liniowy układ równań algebraicznych na przybliżenie U_N^k rozwiązania w chwili t_k,

$$\frac{U_N^k - U_N^{k-1}}{\tau} + P_N U_N^k = f_N^k,$$

gdzie P_N jest określone jak w poprzednim przykładzie oraz

$$f_N^k = \Big(f(x_{ij}, t_k)\Big)$$

i węzły w Ω numerujemy zgodnie z regułą podaną powyżej. Grupując niewiadome po lewej stronie, dostajemy równanie

$$(I + \tau P_N) U_N^k = U_N^{k-1} + \tau f_N^k,$$

a więc takie, w którym macierz układu jest dalej rozrzedzona, ale — dla dostatecznie małych τ — jest „małym zaburzeniem" macierzy jednostkowej.

Gdy siatka *nie jest* regularna[1], macierze dyskretyzacji także tracą regularną strukturę (w powyższych przykładach była ona odwzwierciedlona w tym, że niezerowe elementy macierzy układały się wzdłuż nielicznych diagonali). Macierze bez regularnej struktury są jeszcze trudniejsze dla metod bezpośrednich ze względu na rosnące kłopoty w uniknięciu wypełnienia czynników rozkładu macierzy.

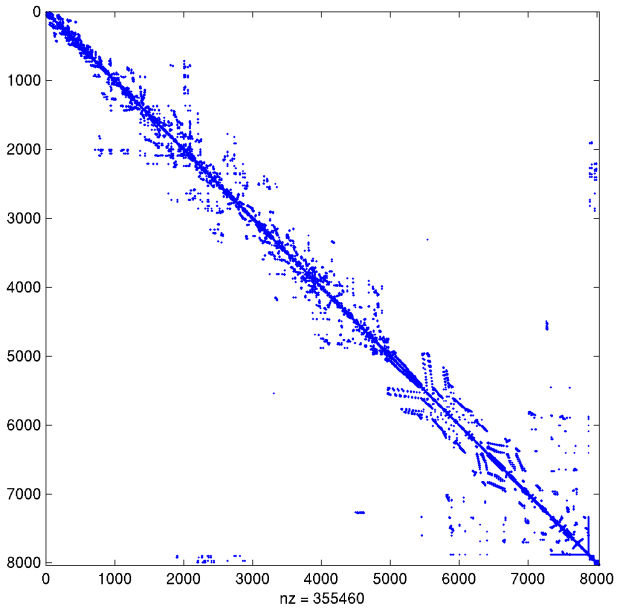
Przykład 5.4 (Macierz z kolekcji Boeinga)**.** Spójrzmy na macierz sztywności dla modelu silnika lotniczego, wygenerowaną swego czasu w zakładach Boeinga i pochodzącą z dyskretyzacji pewnego równania różniczkowego cząstkowego metodą elementu skończonego. Przykład pochodzi z kolekcji Tima Davisa. Jest to mała macierz, wymiaru 8032 (w kolekcji spotkasz równania z milionem i więcej niewiadomych).

Jej *współczynnik wypełnienia* (to znaczy, stosunek liczby niezerowych do wszystkich elementów macierzy) wynosi jedynie 0.006, a więc macierz jest bardzo rozrzedzona: w każdym wierszu są średnio tylko 44 niezerowe elementy.

We wspomnianej kolekcji znajdzuje się jeszcze więcej macierzy rozrzedzonych, pochodzących z najrozmaitszych realnych modeli, nie tylko opartych na równaniach różniczkowych. Choć sporo z nich — tak jak tutaj przedstawiona — będzie symetrycznych, nie jest to regułą; inne układy równań ze zgromadzonego zbioru nie muszą mieć żadnej z pożądanych przez numeryka cech: regularnej/pasmowej struktury, symetrii, ani dodatniej określoności. Jedną z nich przedstawiamy poniżej.

Symulacje działania układów elektronicznych Współczesne układy elektroniczne mogą zawierać miliony części (tranzystorów, rezystorów, kondensatorów), a ich projektowania jest procesem bardzo złożonym i kosztownym. Co więcej, w miarę postępów opracowywania, coraz

[1] Zazwyczaj w zaawansowanych metodach dyskretyzacji równań różniczkowych stosuje się właśnie takie siatki, często — adaptacyjnie dopasowane do przebiegu rozwiązania.



Rysunek 5.2. Struktura niezerowych elementów macierzy bcsstk38.

trudniej o dokonanie w nich poważniejszych zmian projektu. Z tego powodu konieczna jest komputerowa symulacja ich działania. W ogólności prowadzi to do ogromnego układu równań różniczkowo–algebraicznych, ale także bada się modele (prawa Kirchhoffa!), sprowadzające się do rozwiązania układu równań liniowych wielkiego rozmiaru (N na poziomie kilku milionów).

Zobacz ilustrację: *Wizualizacja grafu macierzy Circuit5M o rozmiarze powyżej pięciu milionów*, znajdującą się na stronie WWW przedmiotu.

Duże łańcuchy Markowa Modele wielostanowych systemów kolejkowych (np. routera obsługującego wiele komputerów) także prowadzą do gigantycznych układów równań z macierzami rozrzedzonymi o specyficznej strukturze. Jeśli taki układ może z danego stanu przejść tylko do niewielu innych, macierz reprezentująca łańcuch Markowa będzie rozrzedzona.

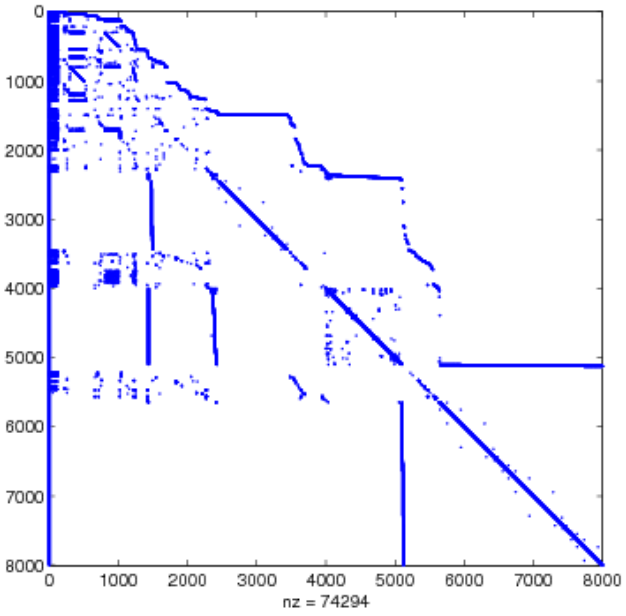
Jednym z zadań, w którym rozważa się bodaj największy „naturalnie" pojawiający się łańcuch Markowa jest zadanie wyznaczania tzw. PageRank — wektora wartościującego strony internetowe w wyszukiwarce Google (zob. artykuł „Miara ważności" K. Diksa w Delcie 8/2008): macierz ta jest oryginalnie bardzo rozrzedzona, całkowicie nieregularna i niesymetryczna, a jej rozmiar jest równy liczbie indeksowanych stron WWW (a więc mniej więcej rzędu 10^9).

Ćwiczenie 5.1. Wykaż, że zadanie $T_N x = b$ można rozwiązać, korzystając z eleminacji Gaussa bez wyboru elementu głównego kosztem $O(N)$.

Ćwiczenie 5.2. Niech macierz A będzie macierzą pasmową rozmiaru $N \times N$, o szerokości pasma $2k$:

$$a_{ij} = 0, \qquad \text{gdy } |i - j| > k.$$

Wykaż, że rozkład LU (bez wyboru elementu głównego) macierzy A można wyznaczyć kosztem $O(Nk^2)$ działań arytmetycznych. Jak zmieni się odpowiedź, gdy trzeba będzie dokonać wyboru elementu głównego w kolumnie?



Rysunek 5.3. Struktura niezerowych elementów macierzy PageRank dla portalu http://www.mimuw.edu.pl.

Ćwiczenie 5.3. Zbadaj koszt rozkładu LU macierzy P_N, w którym wykorzystuje się jedynie informację o tym, że P_N jest macierzą pasmową.

Rozwiązanie. Mamy $N = n^2$ i szerokość pasma jest równa n. Zatem na mocy poprzedniego zadania, koszt jest rzędu $O(n^4)$.

5.2. Macierze rozrzedzone: implementacja

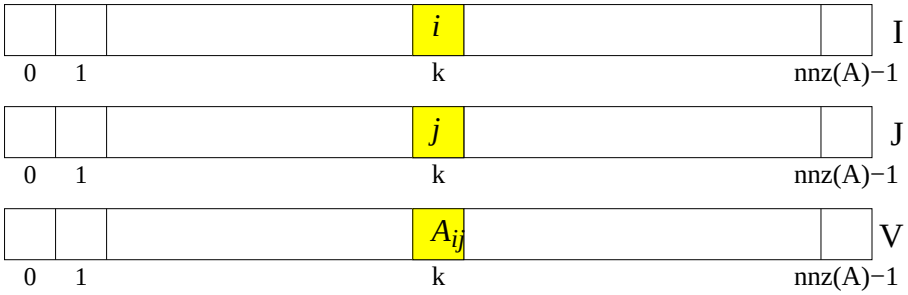
Zanim przystąpimy do omawiania metod rozwiązywania układów równań liniowych z macierzami rozrzedzonymi, warto zapoznać się ze sposobami reprezentacji (formatami) macierzy rozrzedzonych. Ponieważ macierze rozrzedzone mają dużo zerowych elementów, ogólną zasadą jest zapamiętywanie tylko tych różnych od zera, w połączeniu z informacją o ich lokalizacji w macierzy. Spośród wielu struktur danych wygodnych dla przechowywania macierzy rozrzedzonych, opisanych m.in. w monografii [13], największą popularnością wśród numeryków cieszą się dwa: format współrzędnych oraz format spakowanych kolumn (lub wierszy)[2]. Liczbę niezerowych elementów macierzy A rozmiaru $N \times M$ będziemy oznaczali, przez analogię do funkcji MATLABa, $\mathrm{nnz}(A)$.

Format współrzędnych Jest to najprostszy sposób reprezentacji macierzy rozrzedzonych. Do zapamiętania macierzy A rozmiaru $N \times M$, liczącej $\mathrm{nnz}(A)$ niezerowych elementów, wykorzystujemy trzy wektory: I, J — oba typu **int** — oraz V, typu **double**, wszystkie o długości $\mathrm{nnz}(A)$, przy czym zachodzi

$$A_{I[k],J[k]} = V[k], \qquad \forall k = 0, \ldots, \mathrm{nnz}(A) - 1,$$

zob. rysunek 5.4.

[2] Mniej popularny format diagonalny spotkamy m.in. w reprezentacji macierzy pasmowych w LAPACKu.



Rysunek 5.4. Format współrzędnych reprezentacji macierzy rozrzedzonych.

Przykład 5.5 (Macierz jednowymiarowego laplasjanu w formacie współrzędnych)**.** Zdefiniujemy niezbędne struktury danych i następnie wypełnimy je tak, by reprezentować macierz T_N jednowymiarowego laplasjanu (5.5) w formacie współrzędnych. Macierz T_N jest kwadratowa rozmiaru $N \times N$, o co najwyżej $3N$ niezerowych elementach.

Poniższa struktura:

```
typedef struct {
int I;
int J;
double V;
} SpElem;
```

będzie odpowiadała jednemu elementowi reprezentowanej macierzy. Cała macierz T_N zmieści się zatem w tablicy $3N$ elementów typu SpElem, na które przydzielamy miejsce:

```
SpElem *T;

/* alokujemy miejsce na 3N elementów */
T = (SpElem *) calloc(3*N, sizeof(SpElem));
```

Zwróćmy uwagę na to, że do przydzielenia pamięci użyliśmy tym razem funkcji calloc(), a nie malloc(), jak zwykle. To dla większego bezpieczeństwa, bo calloc() wypełnia zerami przydzieloną pamięć.

Teraz możemy wypełnić tablicę. Jak widzimy z dalszego ciągu kodu, zmienną T_NNZ inkrementujemy od początkowej wartości 0 za każdym razem, gdy do macierzy T zapisujemy kolejny element — w ten sposób będziemy dokładnie wiedzieli, ile w rzeczywistości (potencjalnie) niezerowych elementów wpisaliśmy do zaalokowanej tablicy.

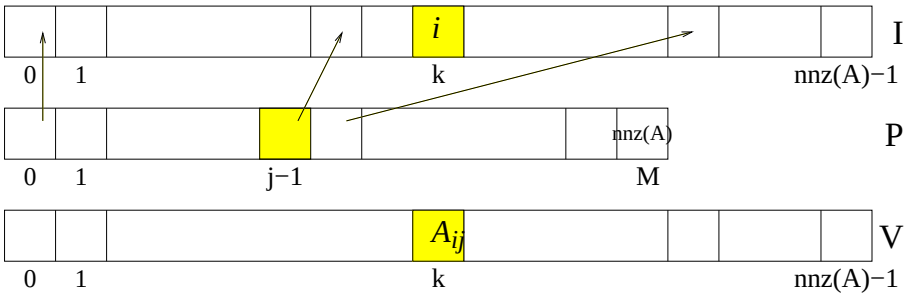
```
int nnz = 0; /* bieżąca liczba wpisanych do tablicy elementów */

/* wypełniamy macierz */
for(i = 1; i <= N−1; i++)
{
  /* element diagonalny */
  T[nnz].I = i;
  T[nnz].J = i;
  T[nnz].V = 2;
  nnz++;
  /* element naddiagonalny w i−tym wierszu */
  T[nnz].I = i;
  T[nnz].J = i+1;
  T[nnz].V = −1;
```

```
  nnz++;
  /* element poddiagonalny w i−tej kolumnie */
  T[nnz].I = i+1;
  T[nnz].J = i;
  T[nnz].V = −1;
  nnz++;
}
/* ostatni element, (N,N) */
T[nnz].I = N;
T[nnz].J = N;
T[nnz].V = 2;
nnz++;
```

Nie zawsze taka forma implementacji formatu współrzędnych będzie możliwa: często wybór implementacji będzie narzucony przez biblioteki, jakich zechcemy używać.

Format spakowanych kolumn (lub wierszy) Format współrzędnych nie narzucał żadnego uporządkowania elementów macierzy — można było je umieszczać w dowolnej kolejności. Z drugiej strony, narzucenie sensownego porządku mogłoby wspomóc realizację wybranych istotnych operacji na macierzy, na przykład, aby wygodnie było realizować działanie (prawostronnego) mnożenia macierzy przez wektor, wygodnie byłoby przechowywać elementy macierzy *wierszami*. Tak właśnie jest zorganizowany format spakowanych wierszy (CSR, ang. *Compressed Sparse Row*). Analogicznie jest zdefiniowany format spakowanych kolumn (CSC, *Compressed Sparse Column)*, którym zajmiemy się bliżej.

Podobnie jak w przypadku formatu współrzędnych, macierz w formacie CSC jest przechowywana w postaci trzech wektorów. V jest wektorem typu **double** o długości nnz(A), zawierającym *kolejne* niezerowe elementy macierzy A wpisywane *kolumnami*, I jest wektorem typu **int**, także o długości nnz(A), zawierającym numery wierszy macierzy A, odpowiadających elementom z V. Natomiast zamiast tablicy J, jak to było w formacie współrzędnych, mamy krótszy wektor typu **int**, P, o długości $M+1$, zawierający na $(j-1)$ miejscu indeks pozycji w V, od której w V rozpoczynają się elementy j-tej kolumny macierzy A (zob. rysunek 5.5).



Rysunek 5.5. Format CSC reprezentacji macierzy rozrzedzonych.

W ten sposób, wartości kolejnych elementów j-tej kolumny macierzy A znajdują się w tablicy V na pozycjach o indeksach P[j−1]... P[j]−1. Konsekwentnie, zawsze P[0]=0, a ostatni element, P[M], jest równy nnz(A). Dzięki temu, takie operacje, jak na przykład wydobycie jednej kolumny z macierzy A, lub mnożenie przez wektor, są łatwiejsze do implementacji w schludnej, bezwariantowej pętli [13].

Ostatecznie mamy więc zależność

$$A_{I[k],j} = V[k], \qquad j = 1, \ldots, M,\ k = P[j-1], \ldots, P[j]-1.$$

Z tego rodzaju formatu reprezentacji macierzy rzadkich (z drobnymi modyfikacjami) korzystają np. pakiety Octave, MATLAB i UMFPACK.

Przykład 5.6 (Procedura mnożenia macierzy w formacie CSC przez wektor)**.** Mając macierz w formacie CSC, zadaną tablicami I[NNZ], P[M+1], V[NNZ], możemy napisać zgrabną procedurę mnożenia jej przez wektor x[M]. Wynik będziemy zapisywać w wektorze y[N]:

```
for(i = 0; i < N; i++)
  y[i] = 0.0; /* zerujemy wektor wynikowy */

for(j = 0; j < M; j++)
  for(k = P[j]; k < P[j+1]; k++)
    y[I[k]] += V[k]*x[j];
```

Ćwiczenie 5.4. Zapisz w pseudokodzie procedurę mnożenia macierzy rozrzedzonej przez wektor, gdy jest ona reprezentowana

1. w formacie współrzędnych,
2. w formacie CSR.

5.3. Metody stacjonarne rozwiązywania układów równań liniowych

Teraz zajmiemy się prostymi i historycznie najstarszymi metodami iteracyjnego rozwiązywania układu równań

$$Ax = b,$$

gdzie nieosobliwa macierz A jest (zapewne wielkiego) rozmiaru N. Są one bodaj najprostsze w analizie i implementacji, ale — jak można się domyślić — w praktyce najmniej efektywne. Z drugiej strony, stanowią one ważny składnik jednej z najszybszych metod rozwiązywania niektórych *trudnych* układów równań (por. rozdział 8.3).

Rozważane przez nas w tym rozdziale metody opierają się na rozkładzie macierzy A na część „łatwo odwracalną", M, i „resztę", $Z = M - A$. Dokładniej, równanie $Ax = b$ można zapisać jako zadanie punktu stałego

$$Mx = Zx + b,$$

a jeśli M jest nieosobliwa, to równoważnie:

$$x = M^{-1}(Zx + b).$$

Ponieważ jest to zadanie znajdowania punktu stałego postaci $x = \Phi(x)$, można spróbować zastosować doń metodę iteracji prostej Banacha:

$$x_{k+1} = M^{-1}(Zx_k + b). \tag{5.12}$$

Takie metody nazywamy *stacjonarnymi metodami iteracyjnymi.*

Ćwiczenie 5.5. Wykaż, że każdą stacjonarną metodę iteracyjną można zapisać w wygodniejszej (dlaczego?) do praktycznej implementacji postaci

$$x_{k+1} = x_k + M^{-1}r_k,$$

gdzie $r_k = b - Ax_k$.

Rozwiązanie.

$$x_{k+1} = M^{-1}(Zx_k + b) = M^{-1}((M - A)x_k + b) = x_k + M^{-1}(b - Ax_k).$$

Jest to wygodniejsza postać, bo wymaga operowania wektorem residuum r_k, na którym zapewne będziemy opierać kryterium stopu iteracji (więc i tak wtedy byśmy go wyznaczali)). Nie wymaga też jawnego Z.

Metody stacjonarne można zapisać w ogólnej postaci

$$x_{k+1} = Bx_k + c, \tag{5.13}$$

gdzie macierz B oraz wektor c są tak dobrane, by punkt stały x^* równania (5.13) był rozwiązaniem równania wyjściowego $Ax = b$.

Dla stacjonarnej metody iteracyjnej, $B = M^{-1}Z = I - M^{-1}A$ oraz $c = M^{-1}b$. Macierz B nazywamy *macierzą iteracji*, gdyż zachodzi

$$x_{k+1} - x^* = B(x_k - x^*) = \ldots = B^{k+1}(x_0 - x^*).$$

Jasne jest, że z twierdzenia Banacha o punkcie stałym wynika od razu następujący warunek zbieżności:

Stwierdzenie 5.1 (warunek wystarczający zbieżności metody stacjonarnej)**.** *Jeśli $\|B\| < 1$, to metoda (5.13) jest zbieżna co najmniej liniowo do x^* dla dowolnego $x_0 \in R^N$. Tutaj $\|\cdot\|$ oznacza dowolną normę macierzową indukowaną przez normę wektorową.*

Dowód. Przy naszych założeniach, mamy do czynienia z kontrakcją $\Phi(x) = Bx + c$, która odwzorowuje kulę $K = \{x \in R^N : \|x - x^*\| \leqslant \|x_0 - x^*\|\}$ w siebie:

$$\|\Phi(x) - x^*\| = \|\Phi(x) - \Phi(x^*)\| = \|Bx - Bx^*\| \leqslant \|B\|\|x - x^*\| < \|x - x^*\|.$$

Ponieważ, z założenia, stała Lipschitza dla Φ wynosi $\|B\| < 1$, to z twierdzenia Banacha o kontrakcji (por. twierdzenie 9.1) wynika teza stwierdzenia. □

Wniosek 5.1. *Jeśli $\|I - M^{-1}A\| < 1$, to ciąg określony przez stacjonarną metodę iteracyjną jest zbieżny do $x^* = A^{-1}b$ oraz*

$$\|x_{k+1} - x^*\| \leqslant \|I - M^{-1}A\|\|x_k - x^*\|.$$

Dowód. Aby stwierdzić zbieżność, wystarczy w poprzednim stwierdzeniu podstawić $B = M^{-1}(M - A) = I - M^{-1}A$ oraz $x = x_k$. Oszacowanie błędu jest konsekwencją własności kontrakcji. □

Warunek konieczny i dostateczny zbieżności tej iteracji dla dowolnego wektora startowego x_0 podaje poniższe twierdzenie. Podkreślmy, że chodzi nam tu o zbieżność metody do rozwiązania układu, gdy startujemy z *dowolnego* przybliżenia początkowego x_0.

Twierdzenie 5.1 (warunek konieczny i dostateczny zbieżności metody stacjonarnej)**.** *Niech $B = I - M^{-1}A$ będzie macierzą iteracji metody (5.13). Ciąg x_k określony tą metodą jest zbieżny do rozwiązania x^* dla dowolnego $x_0 \in R^N$ wtedy i tylko wtedy, gdy*

$$\rho(B) = \max\{|\lambda| : \lambda \text{ jest wartością własną } B\} < 1.$$

Dowód. Na mocy (5.13) otrzymujemy równanie błędu $e_k = x_k - x^*$,

$$e_k = Be_{k-1} = \ldots = B^k e_0.$$

Konieczność warunku jest oczywista: jeśli metoda jest zbieżna dla każdego x_0, to w szczególności dla takiego, że e_0 jest równe wektorowi własnemu macierzy B odpowiadającego zadanej (dowolnej) wartości własnej λ. W konsekwencji,

$$e_k = B^k e_0 = \lambda^k e_0 \to 0,$$

a to oznacza, że $|\lambda| < 1$.

To, że warunek wymieniany w tezie twierdzenia jest także wystarczający dla zbieżności metody, wynika z faktu (zob. [11]), że dla dowolnego $\epsilon > 0$ istnieje norma macierzowa indukowana przez normę wektorową $\|\cdot\|_\epsilon$ taka, że

$$\|B\|_\epsilon \leqslant \rho B + \epsilon.$$

Skoro więc z założenia $\rho(B) < 1$, to można dobrać takie ϵ, by $\|B\|_\epsilon < 1$ i w konsekwencji

$$\|e_k\|_\epsilon \leqslant \|B\|_\epsilon^k \|e_0\|_\epsilon,$$

skąd $x_k \to x^*$. □

Ćwiczenie 5.6. W powyższym dowodzie, jeśli λ jest zespolona, to odpowiadający jej wektor własny też jest zespolony, a więc — formalnie — nie może być uznany za kontrprzykład zbieżności, bo ta ma być „*dla każdego* $x_0 \in \mathbb{R}^N$", a nie — „*dla każdego* $x_0 \in \mathbb{C}^N$". Uzupełnij tę lukę pokazując, że jeśli nawet jeśli λ jest zespolona, to gdy $|\lambda| \geqslant 1$ można wskazać taki *rzeczywisty* wektor x_0, że zbieżność nie będzie zachodzić.

Zaletą stacjonarnych metod iteracyjnych jest ich prostota powodująca, że są one wyjątkow wdzięczne do szybkiego zaprogramowania. Zobaczymy to na przykładzie kilku klasycznych metod stacjonarnych: Jacobiego, Gaussa–Seidela i SOR. Wszystkie będą bazować na podziale macierzy A na trzy części: diagonalną, ściśle dolną trójkątną i ściśle górną trójkątną:

$$A = L + D + U, \tag{5.14}$$

gdzie

$$L = \begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ a_{31} & a_{32} & 0 & & \\ \vdots & \vdots & \ddots & 0 & \\ a_{N1} & a_{N2} & \cdots & \cdots & 0 \end{pmatrix}, \quad D = \begin{pmatrix} a_{11} & & & & \\ & a_{22} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{NN} \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1N} \\ & 0 & a_{23} & \cdots & a_{2N} \\ & & 0 & \ddots & \vdots \\ & & & 0 & \vdots \\ & & & & 0 \end{pmatrix}$$

5.3.1. Metoda Jacobiego

Biorąc w (5.12) $M = D$, gdzie D jest macierzą diagonalną składającą się z wyrazów stojących na głównej przekątnej macierzy A (zob. (5.14)), otrzymujemy (o ile na przekątnej macierzy A nie mamy zera) metodę iteracyjną

$$x_{k+1} = D^{-1}(b - (L + U)x_k),$$

zwaną *metodą Jacobiego.*

Rozpisując ją po współrzędnych, dostajemy układ rozszczepionych równań (numer iteracji wyjątkowo zaznaczamy w postaci *górnego* indeksu):

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right),$$

co znaczy dokładnie tyle, że w i-tym równaniu wyjściowego układu przyjmujemy za współrzędne x wartości z poprzedniej iteracji i na tej podstawie wyznaczamy wartość x_i.

Widzimy więc, że metoda rzeczywiście jest banalna w implementacji, a dodatkowo jest w pełni równoległa: każdą współrzędną nowego przybliżenia możemy wyznaczyć niezależnie od pozostałych.

Twierdzenie 5.2 (O zbieżności metody Jacobiego)**.** *W metodzie Jacobiego warunek dostateczny zbieżności, $\|D^{-1}(L+U)\|_\infty < 1$, jest spełniony np. wtedy, gdy macierz A ma dominującą przekątną, tzn. gdy*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \qquad \forall i = 1, \ldots, N. \tag{5.15}$$

Dowód. Rzeczywiście, ponieważ wyraz (i,j) macierzy $M^{-1}Z$ wynosi 0 dla $i = j$ oraz a_{ij}/a_{ii} dla $i \neq j$, to

$$\|M^{-1}Z\|_\infty = \max_{1 \leqslant i \leqslant N} \sum_{j=1, j \neq i}^{N} |a_{ij}|/|a_{ii}| = \max_{1 \leqslant i \leqslant N} \sum_{j=1}^{N} |a_{ij}|/|a_{ii}| - 1 < 1,$$

przy czym ostatnia nierówność wynika z warunku diagonalnej dominacji. □

Niestety, w wielu ważnych wypadkach metoda Jacobiego, choć zbieżna, będzie zbieżna **zbyt wolno**, by nas zadowolić.

Przykład 5.7 (Macierz laplasjanu)**.** Macierz $N \times N$, zwana *macierzą jednowymiarowego laplasjanu*

$$T_N = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

pojawia się w bardzo wielu zastosowaniach (patrz przykład 5.1), również jako sztucznie wprowadzane podzadanie w niektórych algorytmach numerycznych. Ta macierz jest macierzą taśmową, symetryczną i dodatnio określoną, więc układ równań z tą macierzą można bez trudu rozwiązać metodami bezpośrednimi, kosztem $O(N)$. Jak jednak za chwilę się przekonamy, układ równań z macierzą T_N będzie *wyjątkowo trudny* dla klasycznej metody stacjonarnej.

Stosując do T_N metodę Jacobiego mamy $M = 2I$ oraz $Z = T_N - M$. Obliczając normę macierzy iteracji Jacobiego dostajemy $\|M^{-1}Z\|_\infty = 1$, co — jak wiemy z twierdzenia 5.1 — nie rozstrzyga jeszcze o jej zbieżności lub niezbieżności, ale już może stanowić dla nas poważne ostrzeżenie.

Potrzebna będzie bardziej subtelna analiza. Okazuje się, że są znane wzory na wartości własne λ_j macierzy T_N (por. ćwiczenie 5.8):

$$\lambda_j = 2\left(1 - \cos\left(\frac{j\pi}{N+1}\right)\right) = 4\sin^2\left(\frac{j\pi}{2(N+1)}\right),$$

(a więc, $0 < \lambda_j < 4$) dla $1 \leqslant j \leqslant N$. W konsekwencji, wartościami własnymi $M^{-1}Z = \frac{1}{2}Z = \frac{1}{2}T_N - I$ są liczby $\mu_i = \frac{1}{2}\lambda_i - 1$. Ponieważ $-1 < \mu_i < 1$, znaczy to, że metoda Jacobiego jest zbieżna dla macierzy T_N.

Z drugiej strony, nie dajmy się zwieść optymizmowi matematyka (*„nie martw się, jest zbieżny..."*): nietrudno sprawdzić, że $\rho(M^{-1}Z) = \cos(\pi/(N+1)) = 1 - \dfrac{\pi^2}{2(N+1)^2} + O(N^{-4})$, co oznacza, że metoda Jacobiego — choć zbieżna! — dla dużych N staje się zbieżna tak wolno, że w praktyce bezużyteczna: na przykład, gdy $N = 100$, $\rho(M^{-1}Z) \approx 0.9995$, zatem potrzebowalibyśmy około 4600 iteracji, by mieć pewność, że początkowy błąd zredukowaliśmy zaledwie... 10 razy.

Dopiero w rozdziale 8.3 przekonamy się, jak — przez niebanalne wykorzystanie głębszych informacji o samym zadaniu — można wykorzystać metodę stacjonarną do konstrukcji metody iteracyjnej o optymalnym koszcie, którą będziemy mogli stosować także do macierzy dwu- i trójwymiarowego laplasjanu.

Ćwiczenie 5.7. Wykaż, że wartości własne macierzy dwuwymiarowego laplasjanu P_N — zob. (5.10) — są postaci

$$\lambda_{jk} = 4\left(\sin^2\left(\frac{j\pi}{2(n_x+1)}\right) + \sin^2\left(\frac{k\pi}{2(n_y+1)}\right)\right),$$

dla $j = 1, \ldots, n_x, k = 1, \ldots, n_y$. Podobnie pokaż, że, wartości własne macierzy trójwymiarowego laplasjanu S_N — zob. (5.11) — są postaci

$$\lambda_{jkl} = 4\left(\sin^2\left(\frac{j\pi}{2(n_x+1)}\right) + \sin^2\left(\frac{k\pi}{2(n_y+1)}\right) + \sin^2\left(\frac{l\pi}{2(n_z+1)}\right)\right),$$

dla $j = 1, \ldots, n_x, k = 1, \ldots, n_y, l = 1, \ldots, n_z$.

Rozwiązanie. Niech $T_N = V\Lambda V^T$ będzie rozkładem macierzy jednowymiarowego laplasjanu takim, że Λ jest macierzą diagonalną zawierającą wartości własne, a V taka, że $V^TV = I$ zawiera wektory własne T_N. Ponieważ macierz $P_N = I \otimes T_N + T_N \otimes I$ to konsekwentnie

$$P_N = ...$$

Stąd wynika teza. Analogicznie postępujemy dla macierzy trójwymiarowego laplasjanu.

Ćwiczenie 5.8. Wykaż, że jeśli $b \neq 0$ oraz $c \neq 0$, to rzeczywista trójdiagonalna macierz Toeplitza[3]

$$T = \begin{pmatrix} d & b & & & \\ c & d & b & & \\ & \ddots & \ddots & \ddots & \\ & & c & d & b \\ & & & c & d \end{pmatrix}_{N\times N} \tag{5.16}$$

ma jednokrotne wartości własne, równe

$$\lambda_k = d + 2c\alpha\cos(k\theta), \qquad k = 1, \ldots, N, \tag{5.17}$$

gdzie

$$\alpha = \sqrt{\frac{b}{c}}, \qquad \theta = \frac{\pi}{N+1}.$$

[3] Macierzą Toeplitza nazywamy macierz o stałych współczynnikach na diagonalach.

Zwróć uwagę na to, że gdy $bc < 0$, wartości własne są zespolone. Wektor własny v_k macierzy T odpowiadający λ_k ma współrzędne dane wzorem

$$(v_k)_j = \alpha^{-j} \sin(jk\theta), \qquad j = 1, \ldots, N. \tag{5.18}$$

Wskazówka. Dla $X = \operatorname{diag}(\alpha, \alpha^2, \ldots, \alpha^N)$ *zachodzi*

$$T = dI + c\alpha X^{-1} T_N^0 X = X^{-1} \left(dI + c\alpha T^0 \right) X,$$

gdzie

$$T^0 = \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{pmatrix}_{N \times N}.$$

Znaczy to, że T *jest podobna do macierzy* $dI + c\alpha T^0$ *— a więc ma te same wartości własne. Nietrudno sprawdzić wprost (ze wzorów trygonometrycznych), że* T^0 *ma wektory i wartości własne takie, jak przewiduje wzór.*

Przykład 5.8. Niech B będzie losową kwadratową macierzą $N \times N$, która ma średnio 10 elementów w wierszu (zatem jej współczynnik wypełnienia wynosi $10/N$) i niech

$$A = B + pI.$$

Dobierając do B wartość p możemy sterować stopniem diagonalnej dominacji macierzy A i, ostatecznie, zagwarantować sobie, że A jest diagonalnie dominująca.

W poniższym eksperymencie komputerowym możemy naocznie przekonać się, jak stopień dominacji diagonali wpływa na szybkość zbieżności metody Jacobiego. Dla porównania zobaczymy, jak z naszą macierzą radzi sobie metoda bezpośrednia: oparta na rozkładzie LU macierzy A i starająca się wykorzystać w inteligentny sposób jej rozrzedzenie.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

function [x, info, iter, resid] = jacobi(A, b, tol=1e-8, maxit=200, x0)
%
% metoda Jacobiego na macierzy A
%
if (nargin < 5)
```

To tylko fragment skryptu Octave. Możesz go uruchomić na `http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch5`.

Ćwiczenie 5.9. — Przekonaj się, czy stopień diagonalnej dominacji (wartość parametru `p` w skrypcie) ma istotny wpływ na szybkość zbieżności.

— Domyślnie testy prowadzimy dla macierzy rozmiaru 2500. Sprawdź, jak zmieni się czas wykonania skryptu, gdy N będzie jeszcze większe: która metoda: iteracyjna, czy bezpośrednia zagwarantuje nam sensowny wynik w krótszym czasie?

— Przekonaj się, że gdy rozmiar macierzy jest niewielki, na przykład $N = 100$, nie ma większego sensu stosować metody iteracyjnej. Przy jakim poziomie tolerancji `tol` metoda iteracyjna staje się konkurencyjna dla bezpośredniej (pod względem czasu działania)?

Ćwiczenie 5.10. — Uzupełnij powyższy skrypt o wyznaczenie teoretycznej wartości współczynnika redukcji błędu w normie $\|\cdot\|_\infty$ i porównaj ją z faktyczną szybkością redukcji błędu.
— Sprawdź, że symetria A nie ma większego wpływu na szybkość zbieżności.

Przykład 5.9. Zdarzają się macierze — niestety, nie są to sztucznie generowane, akademickie przypadki — które są patologicznie trudne dla stacjonarnej metody iteracyjnej, a bardzo łatwe dla metody bezpośredniej. Jedną taką macierz już znamy: jest to macierz jednowymiarowego laplasjanu T_N.

W naszym teście, dla zadania z macierzą $A = T_N + pI$ i różnych wartości parametru $p \geqslant 0$ zbadamy szybkość zbieżności metody Jacobiego i porównamy czas jej działania z metodą bezpośrednią (w przypadku macierzy trójdiagonalnej jest ona praktycznie optymalna).

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

function [x, info, iter, resid] = jacobi(A, b, tol=1e-8, maxit=1000, x0)
%
% metoda Jacobiego na macierzy A
%
if (nargin < 5)
```

To tylko fragment skryptu Octave. Możesz go uruchomić na `http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch5`.

Tym razem zmiana p wyraźnie wpływa na szybkość zbieżności, a dla $p = 0$ zbieżność jest — jak już zdążyliśmy się przekonać także analitycznie — jest bardzo wolna.

Ćwiczenie 5.11. — Wyjaśnij przyczynę szybkiej zbieżności metody dla dużych p.
— Domyślnie eksperymenty prowadzimy dla macierzy rozmiaru 25. Sprawdź, jak zmieni się czas wykonania skryptu, gdy N będzie większe: która metoda zagwarantuje nam wynik w krótszym czasie? Jak odpowiedź zależy od p?
— Jeśli zależałoby nam na obliczeniu dokładnego rozwiązania układu, powinniśmy znacząco zmniejszyć warunek stopu metody, `tol`, na przykład do poziomu 10^{-16}. Jak teraz wypada porównanie metody Jacobiego i bezpośredniej, gdy $p = 0$?

Ćwiczenie 5.12. Rozważmy macierz

$$A = \begin{pmatrix} 1 & \alpha & \alpha \\ \alpha & 1 & \alpha \\ \alpha & \alpha & 1 \end{pmatrix}.$$

Wykaż, że
— $A > 0 \iff -1 < 2\alpha < 2$,
— metoda Jacobiego dla tej macierzy jest zbieżna, wtedy i tylko wtedy, gdy $-1 < 2\alpha < 1$.
Czy rezultat o zbieżności metody Jacobiego można uogólnić na przypadek, gdy A rozmiaru $N \times N$ ma na diagonali same jedynki, a poza nią wszystkie jej elementy są równe α?

5.3.2. Metoda Gaussa–Seidela

Heurystyka tej metody[4] opiera się na zmodyfikowaniu metody Jacobiego tak, by w każdym momencie iteracji korzystać z najbardziej „aktualnych" współrzędnych przybliżenia rozwiązania x. Rzeczywiście, przecież wykonując jeden krok metody Jacobiego, czyli rozwiązując kolejno równania skalarne względem $x_i^{(k+1)}$ dla $i = 1, \ldots, N$:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right),$$

nietrudno zauważyć, że w części sumy, dla $j < i$, moglibyśmy odwoływać się — zamiast do „starych" $x_j^{(k)}$ — do „dokładniejszych", świeżo wyznaczonych, wartości $x_j^{(k+1)}$, tzn. ostatecznie wyznaczać

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right).$$

W języku rozkładu macierzy $A = M - Z$ i iteracji $x_{k+1} = M^{-1}(Zx_k + b)$ mielibyśmy więc $M = L + D$ (dolny trójkąt macierzy A z diagonalą) oraz $Z = -U$ (ściśle górny trójkąt A) i konsekwentnie zapis macierzowy iteracji

$$x_{k+1} = (L + D)^{-1}(b - Ux_k).$$

Twierdzenie 5.3 (O zbieżności metody Gaussa–Seidela)**.** *Jeśli macierz A jest diagonalnie dominująca, to metoda Gaussa–Seidela jest zbieżna do x^* dla dowolnego wektora startowego x_0.*

Inny wariant tej metody dostalibyśmy, biorąc za M górny trójkąt macierzy A.

Uwaga 5.1. Obie metody, Jacobiego i (zwłaszcza) Gaussa–Seidela stosuje się także czasem w prostych algorytmach rozwiązywania *układów równań nieliniowych*: ich zaletą jest to, że głównym składnikiem iteracji jest rozwiązywanie *skalarnego* równania nieliniowego na każdym kroku metody.

Metoda Gaussa–Seidela jest w wielu przypadkach rzeczywiście szybciej zbieżna od metody Jacobiego, np. tak jest w przypadku macierzy jednowymiarowego laplasjanu, patrz wniosek 5.2. Wciąż jednak, dodajmy, dla tego zadania jej zbieżność jest zbyt wolna, by ją stosować jako samodzielną metodę.

Przykład 5.10. Kontynuując przykład 5.8, porównamy szybkość zbieżności metody Gaussa–Seidela i Jacobiego na tym samym zadaniu.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, info, iter, resid] = jacobi(A, b, tol=1e-10, maxit=1000, x0)
```

[4] Ciekawostką jest, że Gauss nie miał z nią nic wspólnego, a Seidel był podobno jej przeciwnikiem...

```
%
% metoda Jacobiego na macierzy A
```

To tylko fragment skryptu Octave. Możesz go uruchomić na `http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch5`.

Ćwiczenie 5.13. Porównaj szybkość zbieżności metod Jacobiego i Gaussa–Seidela na macierzy T_N jednowymiarowego laplasjanu.

Wskazówka. Możesz przeprowadzić porównanie, prowadząc dobrze zaplanowane eksperymenty numeryczne. W MATLABie łatwo utworzysz macierz T_N poleceniem

```
e = ones(N,1);
TN = spdiags([-e, 2*e, -e], [-1,0,1], N, N);
```

Jeśli interesuje Cię wynik teoretyczny, czytaj dalej.

5.3.3. Metoda SOR

Zbieżność metody Gaussa–Seidela można przyspieszyć, wprowadzając parametr relaksacji ω i kolejne współrzędne nowego przybliżenia x_{k+1} wyznaczać, kombinując ze sobą poprzednie przybliżenie $x_i^{(k)}$ oraz współrzędną nowego przybliżenia $\tilde{x}_i^{k+1}$, uzyskanego metodą Gaussa–Seidela:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\tilde{x}_i^{k+1}.$$

Gdyby $\omega = 1$, dostalibyśmy z powrotem metodę Gaussa–Seidela. Ponieważ zwykle wybiera się $\omega > 1$, powstałą metodę nazywa się metodą *nadrelaksacji* (ang. *successive overrelaxation*, SOR), która jest na swój sposób szczytowym osiągnięciem wśród metod stacjonarnych. Rozkład macierzy $A = M - Z$ odpowiadający metodzie SOR z parametrem ω jest zadany przez

$$M = \frac{1}{\omega}D + L, \qquad Z = \left(\frac{1}{\omega} - 1\right)D - U.$$

Twierdzenie 5.4 (lemat Kahana o dopuszczalnych wartościach parametru relaksacji SOR)**.** *Jeśli A ma niezerową diagonalę, a metoda SOR jest zbieżna, to musi być $0 < \omega < 2$.*

Dowód. Wystarczy zbadać promień spektralny macierzy iteracji metody SOR, która jest równa $M^{-1}Z = (D+\omega L)^{-1}((1-\omega)D - \omega U)$. Ponieważ macierz $(D+\omega L)$ jest macierzą trójkątną o diagonali D, to $\det(D+\omega L)^{-1} = \det(D)^{-1}$ i w konsekwencji

$$\det(M^{-1}Z) = \det(D+\omega L)^{-1}\det((1-\omega)D-\omega U) = \det D^{-1}\det((1-\omega)D) = \det((1-\omega)I) = (1-\omega)^N.$$

Ze względu na to, że wyznacznik macierzy jest równy produktowi jej wartości własnych, dochodzimy do wniosku, że $\rho(M^{-1}Z) \geqslant |1-\omega|$, co kończy dowód. □

Jako ciekawostkę odnotujmy poniższe

Twierdzenie 5.5 (Ostrowskiego i Reicha)**.** *Jeśli $A = A^T > 0$, to SOR jest zbieżna dla dowolnego $\omega \in (0, 2)$.*

Dowód tego twierdzenia można znaleźć na przykład w [15, rozdział 8.3].

Dla klasy macierzy obejmującej niektóre spotykane w praktycznych zastosowaniach, można wskazać brzemienny w konsekwencje związek promienia spektralnego macierzy iteracji metody SOR i metody Jacobiego.

Definicja 5.1. Rzeczywistą macierz kwadratową A rozmiaru N, o niezerowych elementach na diagonali, nazywamy macierzą *zgodnie uporządkowaną*, jeśli wartości własne macierzy

$$J_\alpha = \alpha\tilde{L} + \frac{1}{\alpha}\tilde{U},$$

gdzie $\tilde{L} = D^{-1}L$, $\tilde{U} = D^{-1}U$, są niezależne od $\alpha \in \mathbb{C} \setminus \{0\}$.

Zauważmy, że $J_{-1} = -\tilde{L} - \tilde{U}$ jest macierzą iteracji metody Jacobiego dla macierzy A.

Szczególnym przypadkiem macierzy zgodnie uporządkowanych są macierze postaci

$$A = \begin{pmatrix} D_1 & U_2 \\ L_1 & D_2 \end{pmatrix}, \tag{5.19}$$

w których D_1, D_2 są nieosobliwymi macierzami *diagonalnymi*.

Rzeczywiście, dla macierzy postaci (5.19) mamy

$$A = L + D + U,$$

gdzie

$$L = \begin{pmatrix} 0 & 0 \\ L_1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & U_2 \\ 0 & 0 \end{pmatrix}$$

i w konsekwencji

$$J_\alpha = \begin{pmatrix} & \frac{1}{\alpha}D_1^{-1}U_2 \\ \alpha D_2^{-1}L_1 & \end{pmatrix} = \begin{pmatrix} I & \\ & \alpha I \end{pmatrix} \underbrace{\begin{pmatrix} & D_1^{-1}U_2 \\ D_2^{-1}L_1 & \end{pmatrix}}_{J_1} \begin{pmatrix} I & \\ & \alpha I \end{pmatrix}^{-1}.$$

Zatem macierz J_α jest podobna do J_1 i w konsekwencji ma te same wartości własne, co J_1 (niezależnie od $\alpha \neq 0$), a więc — jest zgodnie uporządkowana.

Ćwiczenie 5.14. Wykaż, że jeśli A jest macierzą trójdiagonalną (taką, jak na przykład T_N) lub blokowo trójdiagonalną (taką, jak na przykład P_N) to A można poddać takiej permutacji P wierszy i kolumn, że uzyskana macierz PAP^T jest postaci (5.19)[5].

Rozwiązanie. Rzeczywiście, w przypadku macierzy trójdiagonalnej T_N wystarczy wziąć najpierw zmienne o indeksach nieparzystych, a potem — o indeksach parzystych: $P(1, 2, \ldots, N)^T = (1, 3, \ldots, 2, 4, \ldots)^T$. Dla macierzy P_N wybieramy podobnie (tzw. *red-black ordering*).

[5] W rzeczywistości, można wykazać więcej, że nawet bez tej permutacji macierze T_N i P_N są zgodnie uporządkowane.

Twierdzenie 5.6 (Vargi o związku pomiędzy zbieżnością SOR i metody Jacobiego)**.** *Niech A będzie macierzą zgodnie uporządkowaną i niech $\omega > 0$. Wtedy jeśli μ jest wartością własną macierzy iteracji metody Jacobiego, to $-\mu$ też nią jest. Ponadto, jeśli λ jest różną od zera wartością własną macierzy iteracji metody SOR z parametrem relaksacji ω oraz zachodzi*

$$(\lambda + \omega - 1)^2 = \lambda\,(\omega\,\mu)^2$$

to μ jest wartością własną macierzy iteracji Jacobiego. Na odwrót, jeśli μ jest wartością własną macierzy iteracji Jacobiego, to λ zadane powyższym wzorem jest wartością własną macierzy iteracji SOR.

Dowód. Macierz iteracji metody Jacobiego to po prostu

$$-(\tilde{L} + \tilde{U}) = J(-1) = J(1),$$

(ostatnia równość na mocy założenia). Stąd oczywiście wynika część pierwsza tezy.

Macierz iteracji metody SOR ma postać

$$G_\omega = (D + \omega L)^{-1}((1-\omega)D - \omega U) = (I - \omega\tilde{L})^{-1}((1-\omega)I + \omega\tilde{U}).$$

Niech teraz λ będzie wartością własną macierzy iteracji metody SOR,

$$\det(\lambda I - G_\omega) = 0.$$

Ponieważ $\det(I - \omega\tilde{L}) = 1$, mamy równoważnie

$$\begin{aligned} 0 &= \det\left((I - \omega\tilde{L})(\lambda I - G_\omega)\right) = \det\left(\lambda(I - \omega\tilde{L}) - (1-\omega)I - \omega\tilde{U}\right) \\ &= \det\left((\lambda + \omega - 1)I - \lambda\omega\tilde{L} - \omega\tilde{U}\right). \end{aligned}$$

Zatem λ jest wartością własną G_ω wtedy i tylko wtedy, gdy $\dfrac{\lambda + \omega - 1}{\omega}$ jest wartością własną macierzy $\lambda\tilde{L} + \tilde{U}$.

W konsekwencji, jeśli tylko $\lambda \neq 0$, to $\dfrac{(\lambda + \omega - 1)}{\omega\sqrt{\lambda}}$ jest wartością własną macierzy $J(\sqrt{\lambda})$. Na mocy założenia, $J(\sqrt{\lambda})$ ma te same wartości własne, co $J(-1)$, co kończy dowód drugiej części twierdzenia. Dowód części trzeciej pozostawiamy jako ćwiczenie. □

Wniosek 5.2 (O zbieżności metody Gaussa–Seidela)**.** *Jeśli A jest zgodnie uporządkowana i promień spektralny macierzy iteracji metody Jacobiego dla A jest równy $\mu < 1$, to promień spektralny macierzy iteracji metody Gaussa–Seidela jest równy μ^2. Znaczy to, że metoda Gaussa–Seidela jest dwukrotnie szybsza od metody Jacobiego.*

Dowód. Rzeczywiście, na mocy twierdzenia Vargi, każdej niezerowej wartości własnej macierzy SOR odpowiada wartość własna macierzy iteracji Jacobiego μ, spełniająca równanie

$$(\lambda + \omega - 1)^2 = \lambda(\omega\mu)^2.$$

Dla metody Gaussa–Seidela, $\omega = 1$, i w konsekwencji $\lambda = \mu^2$. □

Korzystając z twierdzenia Vargi można udowodnić

Twierdzenie 5.7 (Twierdzenie Younga o optymalnym parametrze SOR)**.** *Niech macierz A będzie symetryczną macierzą zgodnie uporządkowaną i niech $\mu < 1$ będzie równe promieniowi spektralnemu macierzy iteracji metody Jacobiego. Wtedy optymalna wartość parametru relaksacji ω, gwarantująca najmniejszy promień spektralny macierzy iteracji metody SOR, jest równa*

$$\omega_{opt} = \frac{2}{1+\sqrt{1-\mu^2}}.$$

Promień spektralny macierzy iteracji SOR z tym parametrem wynosi wtedy

$$\left(\frac{\mu}{1+\sqrt{1-\mu^2}}\right)^2.$$

Dowód powyższego twierdzenia można znaleźć na przykład w [15, rozdział 8.3].

Jakkolwiek piękne teoretycznie, nawet twierdzenia Younga i Vargi są dość bezlitosne dla metody SOR w wielu praktycznych zastosowaniach.

Przykład 5.11. Dla macierzy jednowymiarowego laplasjanu rozmiaru N, oznaczając $\xi = \pi/(N+1)$, mamy $\mu = \cos(\xi)$ i w konsekwencji $\omega_{opt} = 2/(1+\sin(\xi))$. Zatem, dla $N = 100$, gdy $\mu \approx 0.9995$, promień spektralny macierzy iteracji SOR z optymalnym parametrem relaksacji jest równy aż $(1-\sin(\xi))/(1+\sin(\xi)) \approx 0.9396$, a więc jest praktycznie tak samo blisko jedności, jak w przypadku metody Jacobiego!

Przykład 5.12. Kontynuując przykład 5.9, porównamy szybkość zbieżności metody SOR, Gaussa–Seidela (czyli SOR z parametrem $\omega = 1$) i Jacobiego na tym samym, ekstremalnie trudnym dla metod iteracyjnych zadaniu. Na początek weźmiemy $N = 20$ i zbadamy zależność szybkości zbieżności SOR od wartości ω. Korzystając z wyniku poprzedniego przykładu, będziemy mogli także uruchomić SOR z optymalnym parametrem.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, info, iter, resid] = jacobi(A, b, tol=1e−10, maxit=1000, x0)
%
% metoda Jacobiego na macierzy A
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch5.

Zwróć uwagę na dramatyczną przewagę SOR z optymalnym parametrem nad metodą Gaussa–Seidela. Zbadaj, jak zmienią się wyniki, gdy wyraźnie zwiększysz N, np. do tysiąca.

Ćwiczenie 5.15. Gdy macierz iteracji B metody stacjonarnej

$$x_{k+1} = Bx_k + d$$

jest daleka od normalnej, numeryczna realizacja iteracji w arytmetyce zmiennoprzecinkowej ograniczonej precyzji może napotkać na problemy: w początkowych iteracjach residuum może znacząco rosnąć, co w efekcie może doprowadzić do praktycznej utraty zbieżności.

Zbadaj to na przykładzie–zabawce (por. [**?**, 10.2.3]),

$$B = \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix},$$

gdy $\lambda \in (0.51.0)$. Wyznacz promień spektralny tej macierzy, jej normę, a także wykres normy x_k w zależności od k.

Rozwiązanie. Dla $\lambda \approx 1$ (wyraźnie większych od 1), macierz iteracji B, choć o promieniu spektralnym mniejszym niż 1, ma tę właściwość, że norma jej kolejnych potęg początkowo rośnie.

```
lam = 0.9995; B = [lam 1; 0 lam];
resid=[]; x = [1;1];
for i = 1:100000,
  x = B*x; resid(i) = norm(x);
end;
semilogy(resid);
```

Ćwiczenie 5.16. Metoda Richardsona z parametrem $\tau \in \mathbb{R}$ jest określona wzorem

$$x_{k+1} = x_k + \tau(b - Ax_k).$$

Niech macierz A będzie symetryczna.

1. Sprawdź, że jest to stacjonarna metoda iteracyjna oparta na rozszczepieniu $A = M - Z$, gdzie $M = \dfrac{1}{\tau} I$.
2. Wykaż, że jeśli A jest nieokreślona, dla żadnego wyboru τ nie jest prawdą, że metoda Richardsona jest zbieżna do x^* dla dowolnego x_0.
3. Wykaż, że jeśli A jest dodatnio określona, to metoda Richardsona jest zbieżna dla $0 < \tau < 2/\lambda_{\max}(A)$, a dla
$$\tau_{\text{opt}} = \frac{2}{\lambda_{\max}(A) + \lambda_{\min}(A)},$$
promień spektralny macierzy iteracji $M^{-1}Z$ jest najmniejszy możliwy i równy
$$\rho(M^{-1}Z) = \|M^{-1}Z\|_2 = \frac{\kappa - 1}{\kappa + 1}, \qquad \text{gdzie } \kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}.$$

Ćwiczenie 5.17. Niech $A_{\text{sym}} = \dfrac{A + A^T}{2}$ i niech istnieją stałe $0 < \lambda \leqslant \Lambda$ takie, że dla każdego $x \in \mathbb{R}^N$
$$\lambda x^T x \leqslant x^T A_{\text{sym}} x \qquad \text{oraz} \qquad \|Ax\|_2^2 \leqslant \Lambda x^T A_{\text{sym}} x.$$
Wykaż, że wtedy metoda Richardsona jest zbieżna w normie euklidesowej dla $0 < \tau < 2/\Lambda$. Ponadto, jeśli $\tau = 1/\Lambda$, to współczynnik zbieżności ρ w tej normie można oszacować przez $\sqrt{1 - \dfrac{\lambda}{\Lambda}}$.

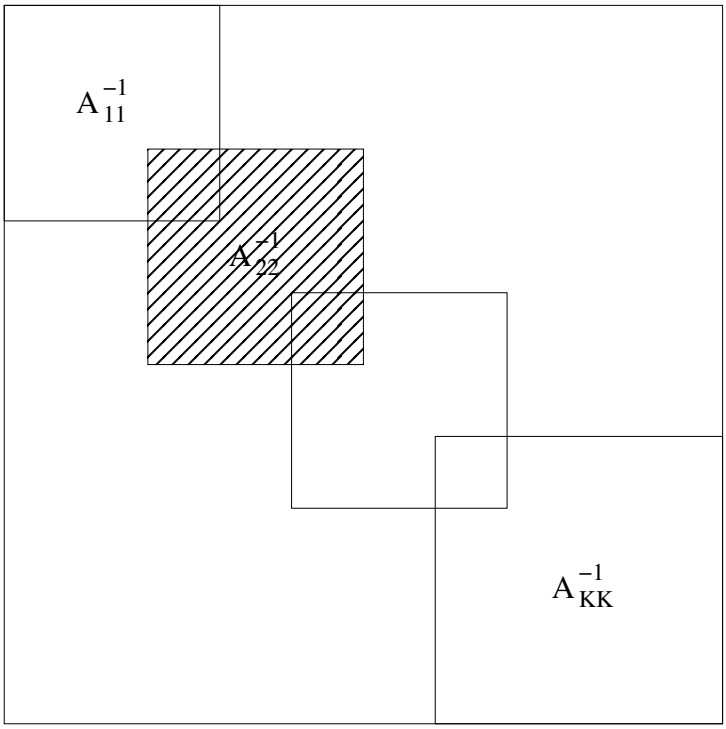
Ćwiczenie 5.18 (Hackbusch)**.** Niech A, M będą macierzami symetrycznymi i dodatnio określonymi. Dla dwóch macierzy symetrycznych X, Y będziemy pisali $X > Y$, jeśli $X - Y > 0$, czyli, innymi słowy, gdy $x^T X x > x^T Y x$ dla każdego $x \neq 0$. Wykaż, że:

1. Jeśli
$$2M > A,$$
to iteracja
$$x_{k+1} = x_k + M^{-1}(b - Ax_k)$$
jest zbieżna, a dokładniej, macierz iteracji, $B = I - M^{-1}A$, spełnia
$$\rho(B) = \|B\|_A = \|B\|_M < 1.$$

2. Jeśli dla pewnych $0 < \lambda \leqslant \Lambda$ zachodzi
$$\lambda W \leqslant A \leqslant \Lambda W,$$
to wszystkie wartości własne $B = I - M^{-1}A$ są rzeczywiste oraz $1 - \Lambda \leqslant \lambda(B) \leqslant 1 - \lambda$.

5.4. Metody blokowe

Pewnym *remedium* na powolną zbieżność metod stacjonarnych jest stosowanie metod blokowych, w których iterację konstruuje się tak jak dotychczas, ale zamiast na elementach, działa się na *całych blokach* macierzy (naturalnie, jeśli trzeba, musimy założyć ich odwracalność). Dla poprawienia zbieżności często stosuje się wariant z zakładką, zob. rysunek 5.6. W praktyce takie metody mogą być wyraźnie skuteczniejsze od dotychczas rozważanych: rzeczywiście, w skrajnym przypadku, pojedynczego „bloku" — rozmiaru N — wszystkie wyżej omówione metody stacjonarne redukują się do metody bezpośredniej...



Rysunek 5.6. Schemat ideowy blokowej metody Jacobiego z zakładką: zamiast standardowego $D^{-1}r = \sum_{i=1}^{N} e_i d_{ii}^{-1} e_i^T r$ wyznaczamy $\sum_{i=1}^{K} R_i A_{ii}^{-1} R_i^T r$, gdzie R_i jest macierzą operatora obcięcia r do współrzędnych odpowiadających A_{ii}.

Ciekawym i nietrywialnym uogólnieniem metod z zakładką jest addytywna metoda Schwarza: jeden z przedstawicieli klasy tzw. metod dekompozycji obszaru, będących bardzo efektywnymi i naturalnie równoległymi operatorami ściskającymi (mówimy o nich w rozdziale 7) dla dyskretyzacji równań różniczkowych cząstkowych, [14].

5.5. Metody projekcji

Dotychczas omówione metody opierają się na czysto formalnym, mechanicznym, podziale macierzy, bezpośrednio związanym z jej strukturą. Tymczasem można zdefiniować metody iteracyjne o podobnym koszcie iteracji, ale w których kolejne przybliżenia rozwiązania będziemy wybierać tak, by w jakimś sensie zminimalizować błąd.

Metodę iteracyjną będziemy definiować w formie aktualizacji poprzedniego przybliżenia,

$$x_{k+1} = x_k + \delta_k.$$

„Idealna poprawka" δ_k, oznaczmy ją δ^*, powinna spełniać więc równanie

$$A\delta^* = r_k,$$

bo wtedy dostalibyśmy $x_{k+1} = x^*$. Ma ona jednak tę wadę, że do jej wyznaczenia musielibyśmy dokładnie *rozwiązać* nasz układ równań (co prawda z nieco inną prawą stroną, ale to bez znaczenia!) — a więc wpadlibyśmy z deszczu pod rynnę! Jednak, gdyby można było *tanio* rozwiązać ten układ równań *w przybliżeniu*, to może w ten sposób udałoby się nam określić *zbieżną* metodę?

Aby określić przybliżone rozwiązanie równania poprawki, najpierw przeformułujemy je nieco. Niech dwie macierze: U i V mają tę własność, że kolumny każdej z nich tworzą bazę $\mathbb{R}^N$. Wtedy rozwiązanie dokładne równania idealnej poprawki, δ^*, możemy reprezentować w bazie V, $\delta^* = Va^*$ dla pewnego $a^* \in \mathbb{R}^N$. Co więcej, z nieosobliwości macierzy U i V wynika, że

$$A\delta^* = r_k \iff U^T AVa^* = U^T r_k.$$

To daje nam pomysł na definicję *przybliżonej* poprawki idealnej: niech U_k i V_k będą macierzami pełnego rzędu, tego samego zadanego z góry rozmiaru $N \times d_k$. Wtedy poprawkę δ_k określimy jako wektor

$$\delta_k = V_k a_k$$

taki, że $a_k \in \mathbb{R}^{d_k}$ spełnia równanie

$$U_k^T AV_k a_k = U_k^T r_k. \tag{5.20}$$

Jest to właśnie *metoda projekcji*.

Ponieważ macierz zredukowana $A_k = U_k^T AV_k$ jest kwadratowa rozmiaru d_k, to δ_k będzie tanie do wyznaczenia, gdy d_k jest niewielkie. Nazwa metody bierze się stąd, że z powyższej definicji wynika, że wektor residualny równania poprawki, $r_k - A\delta_k$ jest prostopadły do podprzestrzeni rozpiętej przez kolumny U_k:

$$U_k^T(r_k - A\delta_k) = 0.$$

Z drugiej strony, δ_k będzie dobrze określone tylko wtedy, gdy macierz zredukowana A_k będzie nieosobliwa — co nie zawsze musi być prawdą, nawet jeśli A jest nieosobliwa (pomyśl o macierzy $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ i $U_k = V_k = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$). Aby zagwarantować sobie odwracalność macierzy zredukowanej, zwykle wybiera się macierz pełnego rzędu V_k i w zależności od własności macierzy A dobiera się macierz U_k:

— Jeśli $A = A^T > 0$, to kładziemy $U_k = V_k$. Wtedy macierz zredukowana $A_k = V_k^T A V_k$ jest symetryczna i dodatnio określona.
— Jeśli A jest tylko nieosobliwa, to kładziemy $U_k = AV_k$. Macierz zredukowana jest postaci $A_k = V_k^T A^T A V_k$, a więc jest macierzą zredukowaną poprzedniego rodzaju, ale dla macierzy równań normalnych, $A^T A$.

Jak możemy się domyślić, metody projekcji są *metodami minimalizacji*, co potwierdza poniższy dwuczęściowy lemat:

Lemat 5.1. *Niech V_k będzie zadaną macierzą $N \times d_k$, pełnego rzędu. Oznaczmy przez $\mathcal{V}_k$ podprzestrzeń rozpiętą przez kolumny V_k.*

— *Jeśli $A = A^T > 0$ oraz $U_k = V_k$, to x_{k+1} określone metodą projekcji (5.20) spełnia*

$$\|x^* - x_{k+1}\|_A \leqslant \|x^* - x\|_A \qquad \forall x \in x_k + \mathcal{V}_k.$$

— *Jeśli A jest nieosobliwa i $U_k = AV_k$, to x_{k+1} określone metodą projekcji (5.20) spełnia*

$$\|b - Ax_{k+1}\|_2 \leqslant \|b - Ax\|_2 \qquad \forall x \in x_k + \mathcal{V}_k.$$

Dowód zostawiamy jako ćwiczenie.

Ćwiczenie 5.19. Udowodnij lemat 5.1.

Wskazówka. Patrz [13] lub dowód twierdzenia 6.1.

5.5.1. Metoda najszybszego spadku

Jednym z bardziej prominentnych przykładów metody projekcji jest metoda najszybszego spadku, działająca w przypadku, gdy macierz A jest symetryczna i dodatnio określona. W tej metodzie wybieramy $U_k = V_k = r_k = b - Ax_k$. Ponieważ wymiar przestrzeni rozpiętej przez kolumny U_k jest równy $d_k = 1$, równanie poprawki upraszcza się do jednego równania skalarnego na $a_k \in \mathbb{R}$,

$$a_k = \frac{r_k^T r_k}{r_k^T A r_k}$$

i w konsekwencji $x_{k+1} = x_k + a_k r_k$.

Nazwa metody wywodzi się stąd, że wektor poprawki w tej metodzie jest proporcjonalny do residuum, które z kolei jest kierunkiem gradientu funkcjonału $\phi(x) = \|x^* - x\|_A^2$ w punkcie x_k:

$$\nabla\phi(x_k) = b - Ax_k = r_k.$$

Twierdzenie 5.8 (o zbieżności metody najszybszego spadku)**.** *W metodzie najszybszego spadku,*

$$\|x^* - x_{k+1}\|_A \leqslant \frac{\kappa - 1}{\kappa + 1}\|x^* - x_k\|_A,$$

gdzie $\kappa = \mathrm{cond}_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$.

Dowód. Łatwo wykazać (por. [13, twierdzenie 5.2]), że jeśli $r_k \neq 0$,

$$\|x^* - x_{k+1}\|_A^2 \leqslant \left(1 - \frac{r_k^T r_k}{r_k^T A r_k}\frac{r_k^T r_k}{r_k^T A^{-1} r_k}\right)\|x^* - x_k\|_A^2.$$

Teza wynika z lematu, którego elegancki dowód, pochodzący od Braessa, można znaleźć w [**?**]:

Lemat 5.2 (Kantorowicza)**.** *Niech $A = A^T > 0$. Wtedy dla dowolnego $x \neq 0$,*

$$\frac{x^T A x}{x^T x} \cdot \frac{x^T A^{-1} x}{x^T x} \leqslant \frac{1}{4}(\sqrt{\kappa} + \frac{1}{\sqrt{\kappa}})^2.$$

□

Ćwiczenie 5.20. Wykaż, że w metodzie najszybszego spadku zachodzi $r_{k+1}^T r_k = 0$.

Rozwiązanie. Ponieważ $x_{k+1} = x_k + a_k r_k$, to

$$r_{k+1} = r_k - a_k A r_k,$$

zatem

$$r_k^T r_{k+1} = r_k^T r_k - a_k r_k^T A r_k = \|r_k\|_2^2 - a_k r_k^T A r_k = 0$$

z definicji a_k.

Przykład 5.13. Kontynuujemy przykład 5.8. Chcąc porównywać trzy metody: Jacobiego, Gaussa–Seidela oraz metodę najszybszego spadku, musimy zadbać o to, żeby były spełnione warunki zbieżności tej ostatniej — a więc, aby macierz A była symetryczna i dodatnio określona. Dlatego, tym razem położymy, dla dodatniego p,

$$A = B^T B + pI.$$

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, info, iter, resid] = jacobi(A, b, tol=1e-8, maxit=100, x0)
%
% metoda Jacobiego na macierzy A
```

To tylko fragment skryptu Octave. Możesz go uruchomić na `http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch5`.

Choć A wydaje się dosyć gęsta, w rzeczywistości wciąż jest macierzą rzadką. Zwróćmy uwagę nie tylko na samą szybkość zbieżności — mierzoną liczbą iteracji — ale także na efektywność metody: ile *czasu* zajmuje wyznaczenie przybliżenia z zadaną dokładnością. Metoda najszybszego spadku niewątpliwie ma najtańszą iterację (a najdroższą — metoda Gaussa–Seidela), co ostatecznie przekłada się na większą efektywność metody najszybszego spadku (pomimo mniejszej szybkości zbieżności).

Ćwiczenie 5.21. Wyjaśnij, dlaczego w powyższym przykładzie, dla małych wartości p, metoda Jacobiego czasem nie jest zbieżna.

Rozwiązanie. Bo macierz A nie musi być diagonalnie dominująca.

Przykład 5.14. Dla macierzy jednowymiarowego laplasjanu, mamy

$$\kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} = \left(\frac{\sin(\pi N/2(N+1))}{\sin(\pi/2(N+1))}\right)^2 = O(N^2),$$

zatem dla $N = 100$ mamy współczynnik redukcji błędu na poziomie $(\kappa - 1)/(\kappa + 1) \approx 0.9995$. Znaczy to dokładnie tyle, że metoda nie nadaje się do rozwiązywania takich zadań, gdy N jest duże. Szybkość zbieżności metody najszybszego spadku jest w naszym przykładzie porównywalna z szybkością metody Jacobiego i gorsza od metody SOR z optymalnym parametrem relaksacji.

5.5.2. Metoda najmniejszego residuum

Gdy o macierzy A wiemy jedynie, że jest nieosobliwa, możemy zastosować *metodę najmniejszego residuum*. W tej metodzie wybieramy $V_k = r_k = b - Ax_k$ oraz $U_k = AV_k = Ar_k$. Równanie poprawki znów upraszcza się do jednego równania skalarnego na $a_k \in \mathbb{R}$,

$$a_k = \frac{r_k^T A^T r_k}{r_k^T A^T A r_k}$$

i w konsekwencji $x_{k+1} = x_k + a_k r_k$.

Twierdzenie 5.9. *Załóżmy, że macierz $A_{\text{sym}} = (A + A^T)/2$ jest dodatnio określona i oznaczmy $\mu = \lambda_{\min}(A_{\text{sym}}) > 0$. Wtedy*

$$\|r_{k+1}\|_2 \leqslant \left(1 - \frac{\mu^2}{\|A\|_2^2}\right)^{1/2} \|r_k\|_2.$$

A_{sym} nazywana jest *częścią symetryczną* macierzy A.

Ćwiczenie 5.22. Przeprowadź dowód twierdzenia o zbieżności metody najmniejszego residuum.

Wskazówka. Zob. dowód twierdzenia 5.3 w [13].

Ćwiczenie 5.23. Przypuśćmy, że umiemy tanio rozwiązywać układy równań z (nieosobliwą) macierzą A rozmiaru $N \times N$. Niech będzie dana macierz

$$\tilde{A} = \begin{pmatrix} A & v \\ u^T & \delta \end{pmatrix},$$

gdzie $u, v \in \mathbb{R}^N$ oraz $\delta \in \mathbb{R}$.

— Wskaż warunek konieczny i dostateczny na to, by macierz $\tilde{A}$ była nieosobliwa.
— Podaj możliwie tani algorytm rozwiązywania układu równań z macierzą $\tilde{A}$.

Wskazówka. $\tilde{A}$ jest nieosobliwa wtedy i tylko wtedy, gdy $\delta - u^T A^{-1} v \neq 0$.

Ćwiczenie 5.24 (wzór Shermana–Morrisona)**.** Przypuśćmy, że umiemy tanio rozwiązywać układy równań z (nieosobliwą) macierzą A rozmiaru $N \times N$. Niech będzie dana macierz

$$\tilde{A} = A - vu^T.$$

gdzie $u, v \in \mathbb{R}^N$.

— Wskaż warunek konieczny i dostateczny na to, by macierz $\tilde{A}$ była nieosobliwa.
— Podaj możliwie tani algorytm rozwiązywania układu równań z macierzą $\tilde{A}$.

Wskazówka. Zauważ, że $\tilde{A}x = b$ *wtedy i tylko wtedy, gdy* $\begin{pmatrix} x \\ t \end{pmatrix}$ *jest rozwiązaniem układu*

$$\begin{pmatrix} A & v \\ u^T & 1 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Patrz także stwierdzenie 11.1.

Ćwiczenia testowe 5.25. Czy metodę iteracyjną warto stosować do rozwiązywania układu $Ax = b$ z macierzą $N \times N$, w przypadku, gdy N jest bardzo duże oraz A jest

1. diagonalna
   NIE. Komentarz do odpowiedzi prawidłowej: *Oczywiście, macierz diagonalną możemy rozwiązać bezpośrednio kosztem N flopów — a więc optymalnym. Wiele metod jednak sprowadziłoby się do metody bezpośredniej — na przykład metoda Jacobiego....* Komentarz do odpowiedzi błędnej: *Zastanów się, ile kosztowałoby rozwiązanie takiego układu równań metodą bezpośrednią, wykorzystującą specjalną postać macierzy.*
2. trójdiagonalna
   NIE. Komentarz do odpowiedzi prawidłowej: *Rozkład LU macierzy trójdiagonalnej możemy wyznaczyć bezpośrednio kosztem $O(N)$ flopów, podobnie z rozwiązaniem samego układu (macierz L jest dwudiagonalna, a macierz U — co najwyżej trójdiagonalna). Stałe przy tym nie są zbyt duże.* Komentarz do odpowiedzi błędnej: *Zastanów się, ile kosztowałoby rozwiązanie takiego układu równań metodą bezpośrednią, wykorzystującą specjalną postać macierzy.*
3. pełna, tzn. bez zerowych elementów
   TAK. Komentarz do odpowiedzi prawidłowej: *Ponieważ macierz jest pełna, to jedno mnożenie wektora przez A kosztuje $O(N^2)$ flopów, więc metoda iteracyjna będzie miała sens, gdy satysfakcjonujące przybliżenie dostaniemy po $k \leqslant N$ iteracjach.* Komentarz do odpowiedzi błędnej: *Rozkład LU macierzy pełnej możemy wyznaczyć bezpośrednio kosztem $O(N^3)$ flopów, co zwykle jest ponad siły współczesnego komputera, gdy N jest bardzo duże.*

Ćwiczenie 5.26. Czy metodę iteracyjną warto stosować do rozwiązywania układu $Ax = b$ z macierzą $N \times N$, w przypadku, gdy N jest bardzo duże oraz A jest rzadka?

Rozwiązanie. Tutaj, jak wynika z naszych dotychczasowych rozważań, odpowiedź jest zniuansowana. Jedno mnożenie wektora przez A kosztuje zapewne $O(N)$ flopów, więc metoda iteracyjna będzie miała sens, gdy satysfakcjonujące przybliżenie dostaniemy po $k \ll N$ iteracjach (a to, jak wiemy, nie zawsze jest gwarantowane). Z drugiej strony, może istnieć skuteczny *reordering* macierzy (czyli zmiana uporządkowania niewiadomych lub równań), pozwalający tanio wyznaczyć rozwiązanie metodą bezpośrednią [13].

6. Metody iteracyjne Kryłowa

W dalszym ciągu przedmiotem naszego zainteresowania będzie układ równań $Ax = b$, którego rozwiązanie oznaczymy (jak zwykle) przez x^*. O ile nie będzie jasno zaznaczone, że jest inaczej, będziemy przyjmować że początkowe przybliżenie rozwiązania, x_0, jest dowolnie zadanym wektorem w R^N (na przykład, $x_0 = 0$).

Tym razem rozważymy pewien wariant metody projekcji, w którym rozmiar podprzestrzeni będzie *powiększał się wraz z postępem iteracji*. Kolejne przybliżenie x_k będziemy dobierać w taki sposób, by $x_k \in x_0 + K_k(r_0, A)$ oraz spełniało pewien dodatkowy warunek, na przykład — minimalizowało pewną miarę błędu na *przestrzeni Kryłowa*

$$K_k(r_0, A) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\},$$

gdzie $r_0 = b - Ax_0$ jest residuum początkowym. (Zwróć uwagę, że przestrzeń Kryłowa jest rozpięta przez kolejne wektory metody potęgowej, o której więcej w rozdziale **??** — to nie przypadek!). Tam, gdzie nie będzie to prowadziło do nieporozumień, będziemy pomijali parametry przestrzeni Kryłowa i pisali po prostu K_k zamiast $K_k(r_0, A)$.

Metody zbudowane zgodnie z powyższym schematem będziemy ogólnie nazywać *metodami Kryłowa*. Jak za chwilę zobaczymy, taki sposób konstrukcji iteracji pozwoli metodzie samodzielnie „dostosować się" do własności macierzy, przez co metody Kryłowa, w przeciwieństwie do metod typu relaksacji, będą w stanie wykorzystać korzystne własności macierzy do przyspieszenia zbieżności: coś, czego ani metody stacjonarne, ani proste metody projekcji nie były w stanie osiągnąć!

Jasne jest, że przestrzenie Kryłowa tworzą wstępujący ciąg podprzestrzeni:

$$K_0 \subseteq K_1 \subseteq \ldots \subseteq K_k \subseteq K_{k+1} \subseteq \ldots \subseteq \mathbb{R}^N.$$

Łatwo pokazać, że nie tylko same przybliżenia x_k, ale także błędy i residua na k-tym kroku metody Kryłowa można powiązać z pewnymi przestrzeniami Kryłowa:

Stwierdzenie 6.1. *Jeśli $x \in x_0 + K_k(r_0, A)$, to $x - x^* \in K_{k+1}(x_0 - x^*, A)$ oraz $b - Ax \in K_{k+1}(r_0, A)$.*

Z tego faktu wynika, że metody przestrzeni Kryłowa są metodami *wielomianowymi*: zarówno błąd, jak i residuum dadzą wyrazić się jako pewien wielomian macierzy A działający na, odpowiednio, błędzie albo residuum początkowym:

Wniosek 6.1. *Jeśli $x \in x_0 + K_k(r_0, A)$, to błąd $x - x^*$ jest postaci $(I + c_1A + \ldots + c_kA^k)(x_0 - x^*)$ i analogicznie residuum $b - Ax$ jest postaci $(I - c_1A - \ldots - c_kA^k)r_0$, gdzie $c_1, \ldots, c_k$ są pewnymi rzeczywistymi współczynnikami.*

Ćwiczenie 6.1. Udowodnij powyższe stwierdzenie i wniosek.

Wskazówka. Wystarczy zauważyć, że $x \in x_0 + K_k(r_0, A)$ jest postaci $x = x_0 + \sum_{i=0}^{k-1} c_{i+1}A^i r_0$.

Konstrukcja dobrej metody Kryłowa powinna więc (jeśli mielibyśmy zachować ducha metod projekcji z poprzedniego rozdziału) zmierzać w stronę wskazania takiego wielomianu, by norma błędu była jak najmniejsza — na każdym kroku iteracji będziemy wtedy musieli rozwiązać

pewne zadanie optymalizacyjne. W zależności od wyboru sposobu miary błędu i warunku optymalności, dostaniemy inną metodę iteracyjną: CG, GMRES, PCR, BiCG i inne. W niniejszym wykładzie omówimy pokrótce tylko najpopularniejsze: CG dla macierzy symetrycznych i dodatnio określonych oraz GMRES dla pozostałych. Należy pamiętać, że w zależności od konkretnego zadania, inne metody mogą okazać się bardziej skuteczne od tutaj omawianych.

W praktycznej implementacji, metody przestrzeni Kryłowa są metodami, w których do ich realizacji musimy jedynie umieć wykonać *mnożenie macierzy przez wektor* — nie jest potrzebne odwoływanie się do poszczególnych elementów lub części macierzy (co było konieczne w metodach opartych na podziale macierzy, takich jak np. metoda Gaussa–Seidela). Metody, które wymagają wykonywania jedynie mnożenia macierzy przez wektor, a nie odwołują się do poszczególnych elementów macierzy, nazywamy **metodami operatorowymi** (ang. *matrix-free methods*). Są one szczególnie pożądane w przypadku, gdy macierz ma nieregularną strukturę i jest bardzo wielkiego rozmiaru, a także w przypadku, gdy sama macierz po prostu nie jest jawnie wyznaczona.

Innym ważnym zastosowaniem metod iteracyjnych Kryłowa jest szybka realizacja jednego kroku tzw. niedokładnej metody Newtona rozwiązywania układu równań nieliniowych, por. rozdział 10.2.4.

Ćwiczenie 6.2. Czy metoda
— Jacobiego
— Richardsona
— najszybszego spadku
jest metodą operatorową?

Na zakończenie wskażemy ciekawą teoretyczną właściwość metod Kryłowa opartych na minimalizacji. Przez $\|x\|_B$ będziemy oznaczali normę energetyczną wektora x, indukowaną przez symetryczną, dodatnio określoną macierz B,

$$\|x\|_B = \sqrt{x^T B x}.$$

Twierdzenie 6.1 (metody Kryłowa oparte na minimalizacji błędu albo residuum w normie energetycznej)**.** *Niech $B \in \mathbb{R}^N$ będzie pewną macierzą symetryczną i dodatnio określoną i niech k-ta iteracja metody Kryłowa, $x_k \in x_0 + K_k$, będzie określona przez jeden z warunków:*

1. *minimalizacji błędu:*

$$\|x_k - x^*\|_B \leqslant \|x - x^*\|_B \qquad \forall x \in x_0 + K_k,$$

albo

2. *minimalizacji residuum:*

$$\|r_k\|_B \leqslant \|b - Ax\|_B \qquad \forall x \in x_0 + K_k.$$

Wtedy metoda jest dobrze określona i znajduje dokładne *rozwiązanie x^* najpóźniej po N iteracjach (jest to tzw. własność* skończonego stopu*).*

Powyższe twierdzenie stosuje się do wielu konkretnych metod Kryłowa znajdujących się w powszechnym użyciu. Przykładowo, w metodzie CG (którą dokładniej omówimy w rozdziale 6.1), zdefiniowanej dla macierzy A symetrycznej i dodatnio określonej, iterację będziemy

określać warunkiem minimalizacji błędu przy $B = A$, natomiast w metodzie GMRES (opisanej w rozdziale 6.2) wybierzemy warunek minimalizacji residuum dla $B = I$.

Część druga powyższego twierdzenia — o *skończonym stopie* metody — ma znaczenie głównie teoretyczne. Wynika to z dwóch przesłanek:

— gdy N jest *bardzo duże*, wykonanie N iteracji, nawet jeśli każda kosztuje tylko $O(N)$ flopów, jest zwykle ponad nasze możliwości obliczeniowe;

— w implementacji metody stosuje się rozmaite chwyty służące maksymalnemu obniżeniu czasochłonności i pamięciożerności algorytmu, które zwykle opierają się na subtelnych zależnościach pomiędzy wyznaczanymi wartościami pośrednimi; gdy algorytm realizujemy w arytmetyce skończonej precyzji, z powodu błędów zaokrągleń tracimy te relacje i z postępem iteracji będą one zwykle coraz gorzej spełnione — co w efekcie doprowadzi także do utraty własności skończonego stopu algorytmu.

Dowód. (twierdzenia 6.1) Rozważmy najpierw przypadek 1, tzn. minimalizacji normy błędu. Pokażemy, że w ogólności x_k jest dobrze określony przez warunek minimalizacji, gdyż jest zdefiniowany jako rozwiązanie pewnego liniowego zadania najmniejszych kwadratów.

Jeśli przez V_k oznaczymy macierz, której kolumnami są wektory tworzące bazę K_k, to każdy $x \in x_0 + K_k$ daje się zapisać w postaci $x = x_0 + V_k a$, gdzie a jest wektorem o tylu współrzędnych, jaki jest wymiar K_k (oznaczmy go $d_k = \dim K_k$). Podstawiając do minimalizowanego wyrażenia, dostajemy zadanie znalezienia $a_k \in \mathbb{R}^{d_k}$ takiego, że

$$\|x_0 + V_k a_k - x^*\|_B \leqslant \|x_0 + V_k a - x^*\|_B \qquad \forall a \in \mathbb{R}^{d_k}; \tag{6.1}$$

(wtedy $x_k = x_0 + V_k a_k$).

Na mocy założenia istnieje macierz $B^{1/2}$ symetryczna i dodatnio określona taka, że $(B^{1/2})^2 = B$ (dlatego nazywamy ją pierwiastkiem z B), a stąd wynika, że $\|x - x^*\|_B = \|B^{1/2}(x - x^*)\|_2$. Znaczy to, że zadanie minimalizacji (6.1) możemy zapisać w równoważnej postaci

$$\|B^{1/2}(x_0 + V_k a_k - x^*)\|_2 \leqslant \|B^{1/2}(x_0 + V_k a - x^*)\|_2 \qquad \forall a \in \mathbb{R}^{d_k}. \tag{6.2}$$

Jest to standardowe zadanie najmniejszych kwadratów,

$$\|M a_k - g\|_2 = \min!,$$

gdzie $M = B^{1/2} V_k$ jest macierzą prostokątną pełnego rzędu, natomiast $g = B^{1/2}(x^* - x_0)$. A zatem x_k jest wyznaczone jednoznacznie.

Na mocy wniosku 6.1, dowolny $x \in x_0 + K_k$ spełnia $x - x^* = p(A)(x_0 - x^*)$, gdzie p jest pewnym wielomianem stopnia co najwyżej k takim, że $p(0) = 1$. Jeśli oznaczymy przez $\tilde{P}_k$ zbiór wielomianów stopnia co najwyżej k o wyrazie wolnym równym 1, to warunek minimalizacji błędu możemy sformułować w równoważnej postaci

$$\|x_k - x^*\|_B = \min_{p \in \tilde{P}_k} \|p(A)(x_0 - x^*)\|_B. \tag{6.3}$$

W szczególności, dla wielomianu $p_N(\lambda) = \det(A - \lambda I)/\det(A)$ — będącego przeskalowanym wielomianem charakterystycznym macierzy A — mamy, że $p_N(0) = 1$ oraz oczywiście p_N jest stopnia nie większego niż N, a więc $p_N \in \tilde{P}_N$. Z drugiej strony, na mocy twierdzenia Cayley'a–Hamiltona, $p_N(A) = 0$, skąd $\|x_N - x^*\|_B = 0$. Znaczy to, że przynajmniej dla $k = N$ zadanie minimalizacji ma (jednoznaczne) rozwiązanie, którym jest $x_N = x^*$ — a więc, że metoda ma własność skończonego stopu.

Dowód drugiej części twierdzenia, dotyczącej przypadku, gdy x_k jest zadany warunkiem minimalizacji residuum, zostawiamy jako ćwiczenie. □

Ćwiczenie 6.3. Uzupełnij powyższy dowód.

Rozwiązanie.

Sposób I Dla dowodu drugiego przypadku zauważmy, że x_k znów jest dobrze określony jako rozwiązanie pewnego liniowego zadania najmniejszych kwadratów postaci

$$\|B^{1/2}(r_0 - AV_k a)\|_2 = \min!$$

oraz na mocy wniosku 6.1 zachodzi

$$\|r_k\|_B = \min_{p \in \tilde{P}_k} \|p(A)(r_0)\|_B. \tag{6.4}$$

Dalszy ciąg dowodu jest identyczny jak poprzednio.

Sposób II Oznaczmy chwilowo macierz indukującą normę energetyczną, w której minimalizujemy residuum, symbolem $\tilde{B}$. Wystarczy skorzystać z pierwszej części twierdzenia, biorąc $B = A^{-T}\tilde{B}A^{-1}$, gdyż

$$\|r_k\|_{\tilde{B}}^2 = r_k^T \tilde{B} r_k = (x^* - x_k)^T A^{-T}\tilde{B}A^{-1}(x^* - x_k) = \|x_k - x^*\|_B.$$

Wniosek 6.2. *Jeśli po k-tej iteracji metody Kryłowa o własności skończonego stopu następuje stagnacja: $K_k = K_{k+1}$, to znaczy, że właśnie znaleziono rozwiązanie dokładne, $x_k = x^*$.*

Dowód. Jeśli $K_k = K_{k-1}$, to oczywiście $x_{k-1} = x_k = x_{k+1} = \ldots = x_N = \ldots$. Ponieważ jednak musi być $x_N = x^*$ (skończony stop po N iteracjach!) to oznacza, że $x_{k-1} = x^*$. □

6.1. Metoda gradientów sprzężonych (CG)

> Algorytm CG został uznany za jeden z 20 najważniejszych algorytmów numerycznych opracowanych w XX wieku.

W niniejszym rodziale będziemy zakładać, że kwadratowa macierz rzeczywista A rozmiaru N jest *symetryczna*, $A = A^T$, oraz jest dodatnio określona,

$$x^T Ax > 0 \quad \forall x \neq 0.$$

Przy tym założeniu, można określić normę energetyczną indukowaną przez A, zadaną tożsamością

$$||y||_A^2 = y^T Ay.$$

Metodę *gradientów sprzężonych*, w skrócie CG (ang. *conjugate gradients*), zdefiniujemy początkowo w sposób niejawny. Kolejne przybliżenie x_k określimy jako wektor z podprzestrzeni afinicznej $x_0 + K_k$, minimalizujący w tej podprzestrzeni błąd w normie energetycznej indukowanej przez A:

$$\|x_k - x^*\|_A \leqslant \|x - x^*\|_A \quad \forall x \in x_0 + K_k. \tag{6.5}$$

Naturalnie, taka definicja może budzić w nas nieco wątpliwości, co do jego *obliczalności* (w sformułowaniu warunku minimalizacji występuje szukane przez nas rozwiązanie *dokładne*, x^*).

Stwierdzenie 6.2 (CG jako metoda bezpośrednia)**.** *Zadanie minimalizacji* (6.5) *ma jednoznaczne rozwiązanie. Jeśli V_k jest macierzą, której kolumny tworzą bazę K_k, to x_k jest dane wzorem $x_k = x_0 + V_k a_k$, gdzie a_k spełnia układ równań*

$$V_k^T A V_k a = V_k^T A(x^* - x_0) = V_k^T r_0. \tag{6.6}$$

Ponadto, w arytmetyce dokładnej, metoda CG znajduje dokładne rozwiązanie w co najwyżej N iteracjach.

Dowód. Jest to natychmiastowy wniosek z twierdzenia 6.1, dla przypadku minimalizacji błędu gdy $B = A$. Zależność (6.6) to nic innego jak układ równań normalnych dla zadania najmniejszych kwadratów (6.2). □

Z powyższego lematu wynika (por. (6.6)), że x_k jest istotnie **obliczalny**: do jego wyznaczenia *nie jest* nam efektywnie potrzebna znajomość rozwiązania!

6.1.1. Implementacja

Aby wyznaczyć x_k, nie będziemy bezpośrednio rozwiązywać układu (6.6) — byłoby to, wraz z postępem iteracji, coraz bardziej kosztowne, ze względu na zwiększający się rozmiar zadania najmniejszych kwadratów. Spróbujemy znaleźć tańszy sposób wyznaczania x_k.

Ponieważ A jest symetryczna, istnieje baza ortogonalna w $\mathbb{R}^N$ złożona z wektorów własnych $q_1, \ldots, q_N$:

$$Aq_i = \lambda_i q_i, \quad i = 1, \ldots, N.$$

Oznaczając przez Q macierz, której kolejne kolumny są wektorami własnymi A,

$$Q = \begin{pmatrix} q_1 & | & q_2 & | & \ldots & | & q_N \end{pmatrix},$$

mamy, że Q jest macierzą ortogonalną, $Q^T Q = I = Q Q^T$, a ponadto A ma rozkład:

$$A = Q \Lambda Q^T,$$

gdzie

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{pmatrix}.$$

Gdyby baza przestrzeni K_k była A–ortogonalna (z powodów historycznych, jej elementy oznaczymy $p_0, \ldots, p_{k-1}$ tak, że $V_k = \begin{pmatrix} p_0 & | & p_1 & | & \ldots & p_{k-1} \end{pmatrix}$), tzn. $p_i^T A p_j = 0$ dla $i \neq j$, to wtedy macierz równań normalnych byłaby diagonalna,

$$V_k^T A V_k = \begin{pmatrix} & \vdots & \\ \ldots & p_i^T A p_j & \ldots \\ & \vdots & \end{pmatrix} = \begin{pmatrix} p_0^T A p_0 & & \\ & \ddots & \\ & & p_{k-1}^T A p_{k-1} \end{pmatrix}.$$

Wtedy kolejną iterację można wyznaczyć z jawnego wzoru:

$$x_k = x_0 + \sum_{i=0}^{k-1} \frac{p_i^T r_0}{p_i^T A p_i} p_i. \tag{6.7}$$

Zatem potrzebna jest nam skuteczna metoda wyznaczania bazy ortogonalnej w przestrzeni K_k.... Oczywiście, ze względu na koszt obliczeniowy i pamięciowy, generowanie i następnie ortogonalizacja oryginalnego zestawu wektorów $\{r_0, Ar_0, \ldots, A^{k-1} r_0\}$ rozpinających K_k nie ma większego sensu. W zamian, wykorzystamy specjalne własności wektorów otrzymywanych w trakcie działania metody.

Lemat 6.1 (o ortogonalności residuów)**.** *Residuum na k-tym kroku, r_k, jest prostopadłe do K_k. Ponadto $r_k \in K_{k+1}$.*

Dowód. Uzasadnienie pierwszej części łatwo wynika z układu równań normalnych (6.6), określającego pośrednio x_k. Rzeczywiście, ponieważ $x_k - x_0 = V_k a_k$, to z (6.6) wynika, że $V_k^T A(x_k - x_0) = V_k^T A(x^* - x_0)$. Upraszczając wyrazy z x_0, dostajemy $V_k^T A(x^* - x_k) = 0$, czyli $V_k^T r_k = 0$.

Druga część wynika natychmiast z faktu, że $x_k \in x_0 + K_k$, skąd $Ax_k \in Ax_0 + AK_k$ i w konsekwencji, odejmując stronami od b, dochodzimy do $r_k \in r_0 + AK_k$. Tymczasem z definicji przestrzeni Kryłowa $r_0 + AK_k \subseteq K_{k+1}$. □

Z powyższego wynika, że jeśli $r_{k-1} \neq 0$, to $K_{k-1} \subset K_k$, a więc dopóki nie trafimy w rozwiązanie dokładne, x^*, kolejne przestrzenie Kryłowa w metodzie CG tworzą ściśle wstępujący ciąg przestrzeni, $K_0 \subset K_1 \subset \ldots K_{k-1} \subset K_k \subseteq \mathbb{R}^N$.

W dalszm ciągu założymy więc, że $r_{k-1} \neq 0$ — a więc, że $x_{k-1} \neq x^*$. Przypuśćmy, że mamy już zadaną bazę A–ortogonalną $\{p_0, \ldots, p_{k-1}\}$ przestrzeni K_{k-1} i znamy x_{k-1}, r_{k-1}. Naszym celem będzie wyznaczenie x_k, r_k oraz p_k. Z zależności (6.7) mamy, że

$$x_k = x_{k-1} + \alpha_k p_{k-1}, \tag{6.8}$$

gdzie

$$\alpha_k = \frac{p_{k-1}^T r_0}{p_{k-1}^T A p_{k-1}}. \tag{6.9}$$

Obkładając (6.8) macierzą A i odejmując obustronnie od b dostajemy dodatkowo zależność rekurencyjną na residua,

$$r_k = r_{k-1} - \alpha_k A p_{k-1}. \tag{6.10}$$

Potrzeba nam jeszcze zależności rekurencyjnej pozwalającej wyznaczyć p_k — ostatni wektor bazy ortogonalnej dla K_{k+1} (wcześniejsze znamy z założenia indukcyjnego). Ponieważ z lematu 6.1 wynika, że $K_{k+1} = \mathrm{span}\{r_k, p_0, \ldots, p_{k-1}\}$, znaczy to, że

$$p_k = r_k + \beta_k p_{k-1} + \gamma_k p_{k-2} + \ldots.$$

Mnożąc skalarnie tę równość przez Ap_{k-1} dostajemy z założenia A-ortogonalności

$$\beta_k = -\frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} \tag{6.11}$$

i podobnie, że γ_k oraz wszystkie następne współczynniki są równe zero (ponieważ z lematu o residuach r_k jest ortogonalne do K_k, a $Ap_j \in K_k$ dla $j \leqslant k-2$). Zatem ostatecznie dostajemy kolejną elegancką zależność rekurencyjną, tym razem na wektory bazy A–ortogonalnej dla K_{k+1}:

$$p_k = r_k + \beta_k p_{k-1}. \tag{6.12}$$

Ponieważ $p_0 = r_0$, tym samym zależności (6.8)—(6.11) stanowią domknięty układ: startując z zadanego x_0, jesteśmy w stanie wyznaczać kolejne przybliżenia.

Okazuje się, że powyższe wzory można jeszcze bardziej wymasować, otrzymując w końcu bardzo zwarty i tani algorytm:

Metoda CG

$r = b - Ax$;

$\rho_0 = \|r\|_2^2$, $\beta = 0$, $k = 1$
while not stop
begin
$p = r + \beta p$
$w = Ap$
$\alpha = \dfrac{\rho_{k-1}}{p^T w}$
$x = x + \alpha p$
$r = r - \alpha w$
$\rho_k = \|r\|_2^2$
$\beta = \dfrac{\rho_k}{\rho_{k-1}}$
$k = k + 1$
end

Jak widać, całą iterację da się wykonać, przechowując w pamięci tylko kilka wektorów (a nie, jak możnaby się obawiać, całą przestrzeń K_k), a najdroższym jej elementem jest mnożenie macierzy przez wektor.

Ćwiczenie 6.4. Opierając się na wzorach (6.8)—(6.11), wyprowadź powyższą postać algorytmu CG.

Rozwiązanie. Najpierw wykażemy, że

$$\alpha_k = \frac{\|r_{k-1}\|_2^2}{p_{k-1}^T A p_{k-1}}. \tag{6.13}$$

Ponieważ z założenia $x_j = x_0 + \sum_{i=0}^{j-1} \alpha_{i+1} p_i$ dla każdego $j \leqslant k$, to (mnożąc obustronnie tę równość przez A i odejmując stronami od b) zachodzi także $r_j = r_0 - \sum_{i=0}^{j-1} \alpha_{i+1} A p_i$. Mnożąc skalarnie tę równość przez p_j i uwzględniając A–ortogonalność kierunków p_i dochodzimy do wniosku, że

$$p_j^T r_j = p_j^T r_0.$$

Z drugiej zaś strony, mnożąc (6.12) skalarnie przez r_j otrzymujemy

$$p_j^T r_j = \|r_j\|_2^2 - \beta_{j-1} p_{j-1}^T r_j = \|r_j\|_2^2,$$

ponieważ r_j jest prostopadłe do K_j, w której zawarty jest wektor p_{j-1}. Ostatecznie więc $p_j^T r_0 = \|r_j\|_2^2$ dla każdego $j \leqslant k$. Biorąc $j = k - 1$, z (6.9) otrzymujemy (6.13).
Teraz wyprowadzimy prostszą reprezentację współczynnika β_k. Z rekurencyjnej zależności pomiędzy residuami (6.10) wynika, że

$$r_k^T r_k = r_k^T r_{k-1} - \alpha_k r_k^T A p_{k-1}.$$

Ponieważ z lematu o ortogonalności residuów $r_{k-1} \in K_k$ oraz r_k jest ortogonalne do K_k, to $r_k^T r_{k-1} = 0$, więc podstawiając do powyższego wzoru uzyskane przed chwilą nowe wyrażenie na α_k dostajemy

$$\|r_k\|_2^2 = -\frac{\|r_{k-1}\|_2^2}{p_{k-1}^T A p_{k-1}} r_k^T A p_{k-1} = \|r_{k-1}\|_2^2 \beta_k.$$

Stąd i z (6.13) już wynika wzór na współczynnik β_k,

$$\beta_k = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2}.$$

Dla dużych N, traktowanie CG jako metody bezpośredniej nie miałoby większego sensu — nie dość, że wykonanie aż N iteracji mogłoby być zadaniem ponad możliwości naszego komputera, to jeszcze dodatkowo algorytm wykorzystuje bardzo specyficzne relacje pomiędzy wektorami, a całość jest przecież w praktyce realizowana w arytmetyce zmiennoprzecinkowej o ograniczonej precyzji, w której te relacje nie zachodzą (w sposób dokładny). Prowadzi to do tego, że w miarę postępu iteracji na przykład wektory p_k są coraz mniej ortogonalne i tym samym metoda nie musi dotrzeć do dokładnego rozwiązania.

Dlatego w praktyce znacznie bardziej właściwe wydaje się potraktowanie metody CG (i innych metod Kryłowa) jako „czystej" metody iteracyjnej i oszacowanie *szybkości redukcji błędu* podobnie, jak czyniliśmy to w przypadku metod stacjonarnych.

Twierdzenie 6.2 (o zbieżności CG jako metody iteracyjnej)**.** *Po k iteracjach metody CG,*

$$||x_k - x||_A \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k ||x_0 - x||_A,$$

gdzie $\kappa = \mathrm{cond}_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$.

Dowód. Skorzystamy z własności (6.3). Zauważmy, że

$$\|p_k(A)(x^* - x_0)\|_A \leqslant \|p_k(A)\|_2 \|(x^* - x_0)\|_A$$

oraz

$$\|p_k(A)\|_2 = \max_{\lambda\in\sigma(A)} |p_k(\lambda)|,$$

zatem wystarczy oszacować wartości wybranego wielomianu $p_k \in \tilde{P}_k$ (nasza norma błędu jest i tak nie większa). Niech $M = \lambda_{\max}(A)$ i $m = \lambda_{\min}(A)$. Jako p_k w (6.3) weźmy przeskalowany k-ty wielomian Czebyszewa,

$$p_k(z) = \frac{T_k\left(\dfrac{M+m-2z}{M-m}\right)}{T_k\left(\dfrac{M+m}{M-m}\right)}.$$

Rzeczywiście, $p_k \in \tilde{P}_k$. Ponadto, ponieważ wielomiany Czebyszewa spełniają zależność

$$|T_k(x)| \leqslant 1 \qquad \text{dla } x \in [-1,1],$$

to

$$\max_{z\in[m,M]} |p_k(z)| \leqslant \frac{1}{T_k\left(\dfrac{M+m}{M-m}\right)} = \frac{1}{T_k\left(\dfrac{\kappa+1}{\kappa-1}\right)}.$$

Należy więc oszacować $T_k\left(\dfrac{\kappa+1}{\kappa-1}\right)$. Ponieważ $\dfrac{\kappa+1}{\kappa-1} > 1$, to skorzystamy ze wzoru

$$T_k(x) = \frac{1}{2}(x+\sqrt{x^2-1})^k + \frac{1}{2}(x-\sqrt{x^2-1})^k \qquad \text{dla } |x| \geqslant 1.$$

W szczególności więc, biorąc $x = \dfrac{\kappa+1}{\kappa-1}$ mamy

$$T_k(x) \geqslant \frac{1}{2}(x+\sqrt{x^2-1})^k = \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k.$$

□

Stwierdzenie 6.3. *Jeśli macierz A ma m różnych wartości własnych, to metoda CG w arytmetyce dokładnej znajdzie rozwiązanie dokładne x^* w co najwyżej m iteracjach.*

Ćwiczenie 6.5. Udowodnij powyższe stwierdzenie.

Wskazówka. *Rozważ odpowiednio przeskalowany wielomian* $p(\lambda) = (\lambda - \lambda_1) \cdots (\lambda - \lambda_m)$.

Ćwiczenia testowe 6.6. Wskaż lepszą metodę rozwiązywania układu równań ze źle uwarunkowaną dodatnio określoną macierzą symetryczną:

1. najszybszego spadku
   NIE. Komentarz do odpowiedzi prawidłowej: *Szybkość zbieżności metody najszybszego spadku jest proporcjonalna do współczynnika uwarunkowania macierzy.* Komentarz do odpowiedzi błędnej: *Zastanów się, jak szybkość zbieżności metody zależy od współczynnika uwarunkowania.*
2. gradientów sprzężonych
   TAK. Komentarz do odpowiedzi prawidłowej: *Szybkość zbieżności metody CG jest proporcjonalna do pierwiastka współczynnika uwarunkowania macierzy.* Komentarz do odpowiedzi błędnej: *Zastanów się, jak szybkość zbieżności metody zależy od współczynnika uwarunkowania.*

Przykład 6.1. Kontynuujemy przykład 5.13. Chcąc porównywać cztery metody: Jacobiego, SOR, metodę najszybszego spadku oraz sprzężonych gradientów, będziemy korzystać z macierzy

$$A = B^T B + pI,$$

gdzie $p > 0$ oraz B jest losową macierzą rozrzedzoną. Zwiększanie parametru p nie tylko poprawia diagonalną dominację, ale także poprawia uwarunkowanie A. Jako parametr relaksacji dla SOR wybraliśmy (strzelając w ciemno) $\omega = 1.3$.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, info, iter, resid] = jacobi(A, b, tol=1e-8, maxit=100, x0)
%
% metoda Jacobiego na macierzy A
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch6.

Zwróćmy uwagę na wyraźną przewagę metody CG nad pozostałymi. Sprawdź, czy podobnie jest dla większych wartości N.

Ćwiczenie 6.7. Sprawdź w przykładzie 6.1, czy faktycznie uwarunkowanie macierzy A wpływa na szybkość zbieżności metody CG i najszybszego spadku. Aby zbadać uwarunkowanie macierzy, możesz skorzystać z polecenia **cond**(A), albo wykorzystać estymator uwarunkowania dostępny w pcg.

Ćwiczenie 6.8. Sprawdź, modyfikując kod przykładu 6.1, czy jeśli A nie będzie symetryczna (lub nie będzie dodatnio określona), wpłynie to istotnie na szybkość zbieżności metody CG i najszybszego spadku. Wypróbuj m.in. $A = B + pI$ dla $p > 0$ (brak symetrii) tak dobranego, by $A_{\text{sym}} > 0$ oraz $A = B^T B + pI$ dla $p < 0$ takiego, żeby A miało i dodatnie, i ujemne wartości własne.

Przykład 6.2. Chcąc porównywać cztery metody: Jacobiego, SOR, metodę najszybszego spadku oraz sprzężonych gradientów dla macierzy jednowymiarowego laplasjanu T_N. Jako parametr relaksacji dla SOR wybraliśmy wartość optymalną, zgodnie z przykładem 5.11.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, info, iter, resid] = jacobi(A, b, tol=1e−10, maxit=1000, x0)
%
% metoda Jacobiego na macierzy A
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch6.

Zwróćmy uwagę na wyraźną przewagę metody CG nad pozostałymi metodami iteracyjnymi. Jednak i tak nie wytrzymuje ona konkurencji z metodą bezpośrednią. Ta sytuacja dramatycznie zmieni się, gdy będziemy rozważali dyskretyzacje dwu- lub trójwymiarowego operatora Laplace'a. O ile wtedy metoda CG wciąż ma kłopoty z szybką zbieżnością, to metoda bezpośrednia (typu rozkładu LU) staje się całkowicie bezużyteczna.

6.2. Metoda GMRES

Metoda GMRES (ang. *Generalized Minimum RESidual*) nie wymaga ani symetrii, ani dodatniej określoności macierzy, jest więc bardziej uniwersalna, choć też w realizacji bardziej kosztowna od CG. Jej szczegółowe omówienie, w tym — wyprowadzenie subtelniejszych oszacowań szybkości zbieżności — wykracza niestety poza ramy niniejszego wykładu (zainteresowani mogą skonsultować m.in. podręcznik [13]).

W tej metodzie, przybliżenie na k-tej iteracji ma minimalizować, w przestrzeni afinicznej $x_0 + K_k$, residuum (stąd nazwa) mierzone w normie euklidesowej:

$$\|r_k\|_2 = \|b - Ax_k\|_2 \leqslant \|b - Ax\|_2 \quad \forall x \in x_0 + K_k. \tag{6.14}$$

Stwierdzenie 6.4 (GMRES jako metoda bezpośrednia)**.** *Zadanie minimalizacji* (6.14) *ma jednoznaczne rozwiązanie. Jeśli V_k jest macierzą, której kolumny tworzą bazę K_k, to x_k jest dane wzorem $x_k = x_0 + V_k a_k$, gdzie a_k jest rozwiązaniem zadania najmniejszych kwadratów względem $a \in \mathbb{R}^{\dim K_k}$,*

$$\|r_0 - AV_k a\|_2 = \min! \tag{6.15}$$

Ponadto, w arytmetyce dokładnej, metoda GMRES znajduje rozwiązanie x^* w co najwyżej N iteracjach.

Dowód. Jest to bezpośredni wniosek z twierdzenia 6.1, dla przypadku minimalizacji residuum gdy $B = I$. □

Twierdzenie 6.3 (o zbieżności metody GMRES jako metody iteracyjnej)**.** *Jeśli macierz A jest diagonalizowalna (to znaczy $A = X\Lambda X^{-1}$ i Λ jest macierzą diagonalną[a]), to k-ta iteracja GMRES spełnia*

$$\|r_k\|_2 \leqslant \operatorname{cond}_2(X) \min_{p\in\tilde{P}_k} \max_{\lambda\in\sigma(A)} |p(\lambda)|.$$

[a] Oczywiście, na diagonali Λ znajdują się wartości własne macierzy A. W ogólności więc, zarówno X, jak i Λ mogą być zespolone.

Dowód. Z (6.4) wynika, że

$$\|r_k\|_2 \leqslant \min_{p\in\tilde{P}_k} \|p(A)\|_2 \, \|r_0\|_2,$$

więc wystarczy oszacować $\|p(A)\|_2$:

$$\|p(A)\|_2 = \|p(X\Lambda X^{-1})\|_2 = \|Xp(\Lambda)X^{-1}\|_2 \leqslant \|X\|_2\|p(\Lambda)\|_2\|X^{-1}\|_2 = \operatorname{cond}_2(X) \max_{\lambda\in\sigma(A)} |p(\lambda)|.$$

□

Powyższe twierdzenie jest mało praktyczne, ze względu na to, że zazwyczaj nie znamy macierzy X. Ale jeśli na przykład macierz A jest normalna, to znaczy $A^TA = AA^T$, to wtedy $X^{-1} = X^T$ i w konsekwencji $\operatorname{cond}_2(X) = 1$.

Ćwiczenie 6.9. Niech A będzie macierzą diagonalizowalną. Wykaż, że jeśli A ma m różnych wartości własnych, to GMRES w arytmetyce dokładnej osiągnie rozwiązanie w co najwyżej m krokach.

Rozwiązanie. Wynika z poprzedniego twierdzenia, wystarczy wziąć $p(z) = (z - \lambda_1)\cdots(z - \lambda_m)/\lambda_1\cdots\lambda_m$.

6.2.1. Implementacja

Dla ostatecznej skuteczności metody iteracyjnej ważna jest także jej efektywna implementacja. Załóżmy więc, że z powodzeniem wykonaliśmy $k-1$ kroków metody GMRES i teraz chcemy wyznaczyć x_k. Oznaczamy, jak zwykle, przez V_k macierz, której kolumny tworzą bazę K_k. Wtedy, jeśli nie osiągnięto jeszcze rozwiązania, to $\dim K_k = k$, zatem $V_k \in \mathbb{R}^{N\times k}$. Aby wyznaczyć współczynniki a_k reprezentacji x_k w bazie V_k, $x_k = x_0 + V_ka_k$, musimy rozwiązać względem $a \in \mathbb{R}^k$ liniowe zadanie najmniejszych kwadratów,

$$\|r_0 - AV_ka\|_2 = \min! \tag{6.16}$$

Wydawać by się mogło, że sprawa jest tu prosta, bo AV_k możemy w miarę łatwo wyznaczyć, a potem należałoby skorzystać z jednej z metod rozwiązywania standardowego zadania najmniejszych kwadratów. Jednak jeśli uwzględnić drobny fakt, że do wyznaczenia x_k potrzebna nam będzie znajomość nie samego a_k, ale także V_k, sytuacja robi się nieco niezręczna. Dlatego po raz kolejny spróbujemy spreparować bazę V_k i jednocześnie wykorzystać fakt, że AV_k rozpina podprzestrzeń w K_{k+1} (rozpiętej przez V_{k+1}).

Zwróćmy uwagę na to, że jeśli $\{v_1, \ldots, v_j\}$ są bazą ortogonalną w K_j dla $j = 1, \ldots, k$, to aby rozszerzyć ją do K_{k+1} wystarczy zortogonalizować — zamiast A^kr_0 — wektor Av_k. Stosując zmodyfikowany algorytm ortogonalizacji Grama-Schmidta do Av_k:

Metoda Arnoldiego wyznaczania bazy ortogonalnej

$v_1 = r_0/\|r_0\|_2$
for $j = 1$ **to** $k-1$ **begin**
  $h_{ij} = v_i^T A v_j$, dla $i = 1, \ldots, j$
  $v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i$ {*Aw_j to wektor, który będziemy ortogonalizować*}
  $h_{i+1,j} = \|v_{j+1}\|_2$
  $v_{j+1} = v_{j+1}/h_{i+1,j}$ {*unormowanie v_{j+1}*}
end

dostajemy wprost z relacji $Av_k = \sum_{i=1}^{j+1} h_{ij} v_i$ związek:

$$AV_k = V_{k+1} H_k, \tag{6.17}$$

gdzie H_k jest macierzą Hessenberga rozmiaru $(k+1) \times k$:

Opisana tu zmodyfikowana metoda Grama–Schmidta ortogonalizacji bazy przestrzeni Kryłowa nosi nazwę metody Arnoldiego.

Jest to istotny postęp na drodze ku efektywnemu rozwiązaniu zadania minimalizacji (6.16), gdyż podstawiając doń (6.17) mamy, że

$$\|r_0 - AV_k a\|_2 = \|r_0 - V_{k+1} H_k a\|_2 = \|V_{k+1}^T r_0 - H_k a\|_2$$

— w ostatniej równości skorzystaliśmy z ortogonalności V_{k+1}. Ale przecież $r_0 = \beta v_1$ (gdzie $\beta = \|r_0\|_2$), zaterm $V_{k+1}^T r_0 = (\beta, 0, \ldots, 0)^T = \beta e_1$! Tak więc, zadanie najmniejszych kwadratów (6.16) dla macierzy rozmiaru $N \times k$ sprowadziliśmy do zadania najmniejszych kwadratów:

$$\|\beta e_1 - H_k a\|_2 = \min! \tag{6.18}$$

dla macierzy Hessenberga H_k, rozmiaru $(k+1) \times k$ (a więc: prawie–trójkątnej!)[1].

Stąd dostajemy bazową wersję implementacji algorytmu GMRES:

Metoda GMRES, wersja bazowa

$r_0 = b - Ax_0$, $V_1 = r_0/\|r_0\|_2$, k = 0
while not stop
  {*wykonaj ortogonalizację metodą Arnoldiego*}
    $h_k = V_k^T A v_k$
    $v_{k+1} = Av_k - V_k h_k$
    $h_{k+1,k} = \|v_{k+1}\|_2$
    $v_{k+1} = v_{k+1}/h_{k+1,k}$
  {*rozszerz bazę V_k i macierz H_k*}
    $V_{k+1} = \begin{pmatrix} V_k & v_{k+1} \end{pmatrix}$
    $H_k = \begin{pmatrix} H_{k-1} & h_k \\ & h_{k+1,k} \end{pmatrix}$
  {*znajdź minimum zadania najmniejszych kwadratów*}
    $r_{k+1} = \min_a \|\beta e_1 - H_k a\|_2$
  {*zwiększ licznik iteracji*}
    $k = k + 1$
end
{*rozwiąż ostatnie zadanie najmniejszych kwadratów*}
$a_{k-1} = \arg\min_a \|\beta e_1 - H_{k-1} a\|_2$
$x_k = x_0 + V_{k-1} a_{k-1}$ {*dopiero teraz musimy wyznaczyć przybliżone x*}

Zwróćmy uwagę na pewien drobiazg obniżający koszt iteracji: dopóki nie wyznaczymy rozwiązania z żądaną dokładnością (co możemy określić jedynie poprzez analizę residuum, r_k), nie

[1] Macierz Hessenberga można — przy pomocy (ortogonalnych) przekształceń Givensa — doprowadzić liniowym kosztem do postaci górnej trójkątnej, zob. [9, rozdział 3.5].

musimy znajdować ani a_k, ani x_k. Dzięki temu, koszt jednej iteracji GMRES jest tylko rzędu $O(kN)$ flopów (gdybyśmy wyznaczali a_k na każdej iteracji, dodawałoby to dodatkowe $O(k^2)$ flopów).

Powyższy algorytm w praktyce jest poddawany pewnym drobnym modyfikacjom, związanym z tym, że choć zmodyfikowany algorytm Grama–Schmidta ma numeryczne własności wyraźnie lepsze od klasycznego, to wciąż w realizacji w arytmetyce skończonej precyzji będzie tracił ortogonalność kolumn V_k. Dlatego uzupełnia się go o warunkową reortogonalizację V_k: szczegóły opisane są w [9] (tam także są przykłady ukazujące wagę tego dodatkowego kroku) oraz w [13].

Ćwiczenie 6.10. Przeanalizuj zbieżność metody GMRES dla macierzy kwadratowej

$$A = \begin{pmatrix} \alpha I & X \\ 0 & \beta I \end{pmatrix},$$

gdzie $\alpha, \beta \in \mathbb{R} \setminus \{0\}$, a X jest macierzą $n \times m$.

6.2.2. Wersja z restartem: GMRES(m)

Pewną wadą metody GMRES jest konieczność zapamiętania na k-tym kroku wszystkich wektorów bazy ortogonalnej przestrzeni Kryłowa K_k. Znaczy to, że nakład obliczeń i, co gorsza, ilość pamięci potrzebnej na wykonanie jednego kroku metody rosną wraz z liczbą iteracji. Aby zniwelować tę przypadłość, zwykle stosuje się restarty metody GMRES co m (rzędu kilkunastu–kilkudziesięcu) iteracji: po m iteracjach przerywane są obliczenia, a wyznaczone w ten sposób rozwiązanie przybliżone jest używane jako początkowe przybliżenie do kolejnego wywołania metody. W ten sposób co prawda spowalnia się nieco szybkość zbieżności samej metody, ale za to utrzymujemy w ryzach jej zasobożerność: to często jest rozsądny kompromis.

7. Ściskanie macierzy (preconditioning)

Zbieżność wszystkich dotychczas przez nas poznanych metod iteracyjnych projekcji i Kryłowa zależy od własności spektralnych macierzy rozwiązywanego układu. Pojawiające się w zastosowaniach macierze często mają niestety niekorzystne własności spektralne (np. bardzo duży wskaźnik uwarunkowania), przez co metody iteracyjne zbiegają na nich bardzo wolno.

Przykładowo, dla wielokrotnie tu przywoływanej macierzy d-wymiarowego laplasjanu dyskretyzowanego na siatce o $N = n^d$ węzłach, jej współczynnik uwarunkowania rośnie tak szybko z rozmiarem macierzy (jak n^2), że czyni bezsensownym iteracyjne rozwiązywanie takich układów gdy n jest bardzo duże. . Zgodnie z twierdzeniem o zbieżności CG, musielibyśmy wykonać rzędu $O(n)$ iteracji, by dostatecznie zredukować błąd. Jasne jest, że jeśli $n = 10000$, to koszt iteracji staje się zaporowy, a w międzyczasie dodatkowo mogą odezwać się problemy związane z realizacją iteracji w arytmetyce skończonej precyzji.

Dlatego bardzo korzystna może być wstępna transformacja układu

$$Ax = b$$

z macierzą o niekorzystnych własnościach, do równoważnego układu

$$MAx = Mb, \tag{7.1}$$

którego macierz MA ma znacznie korzystniejsze własności z punktu widzenia używanej metody iteracyjnej.

Okazuje się, że w m.in. przypadku macierzy laplasjanu i podobnych, można faktycznie wskazać taką macierz M, że iteracja nie jest wiele droższa, ale za to znacznie szybciej zbieżna. Z drugiej strony, w innych ważnych z punktu widzenia zastosowań problemach, takiej dobrej macierzy M wciąż nie umiemy wskazać.

W przypadku macierzy symetrycznych widzieliśmy, że kluczowe znaczenie dla zbieżności metody miało rozmieszczenie wartości własnych na osi liczbowej: jeśli wartości własne były bardzo rozrzucone po prostej, to uwarunkowanie było bardzo duże i w konsekwencji zbieżność powolna. Aby zbieżność była szybsza, kluczowe jest, by wartości własne MA były ciasno upakowane w kilku klastrach (najlepiej — w jednym).

Jeśli więc chcielibyśmy przekształcić macierz tak, by metoda iteracyjna dla MA zbiegała szybko, musimy w jakiś sposób „ścisnąć" spektrum macierzy A. Taką operację nazywamy więc *ściskaniem* (ang. *preconditioning* — bo zmiana uwarunkowania), a macierz M — *macierzą ściskającą*.

Aby całość miała sens, macierz ściskająca M powinna być tak dobrana, by jednocześnie zachodziło kilka nieco przeciwstawnych warunków:

— macierz MA powinna mieć znacznie korzystniejsze własności z punktu widzenia szybkości zbieżności (i ostatecznego kosztu) używanej metody iteracyjnej,
— macierz M powinna być łatwa w „konstrukcji"[1],

[1] W metodach operatorowych (*matrix–free*) nie jest nam potrzebna macierz M, tylko sposób obliczania $M \cdot x$, więc to ta operacja w praktyce wymaga „konstrukcji". Doskonałym przykładem byłoby $M = B^{-1}$, gdzie $B \approx A$; wtedy $M \cdot x$ to obliczenie rozwiązania układu równań z macierzą B, więc „konstrukcją" M byłoby znalezienie rozkładu LU macierzy B.

— macierz M powinna być tania w mnożeniu przez wektor (głównym elementem każdej metody iteracyjnej jest mnożenie macierzy przez wektor: tutaj mamy $M \cdot (A \cdot x)$).

Kilka ekstremalnych propozycji na macierz ściskającą to $M = I$ (łatwa w konstrukcji i tania w mnożeniu, ale niestety nic nie polepsza...) oraz $M = A^{-1}$ (rewelacyjnie poprawia szybkość zbieżności metody iteracyjnej, dając zbieżność w jednej iteracji, ale jest droga w mnożeniu i bardzo droga w konstrukcji). Ponieważ obie skrajności są — ze zrozumiałych względów — nie do zaakceptowania, widać więc, że należy poszukiwać czegoś pośredniego: czegoś, co niskim kosztem przybliża (w jakimś sensie) działanie macierzy odwrotnej.

Uogólnieniem *ściskania lewostronego* (7.1) jest *ściskanie obustronne*:

$$M_L A M_R y = M_L b, \tag{7.2}$$

$$x = M_R y. \tag{7.3}$$

(jeśli powyżej $M_L = I$, to ściskanie jest oczywiście tylko prawostronne).

7.1. Proste operatory ściskające

Jednym z powszechniej stosowanych (aczkolwiek nie zawsze dostatecznie skutecznych) sposobów ściskania są te oparte na zastosowaniu jednego kroku klasycznej metody iteracyjnej (lub jej wersji blokowej). Ponieważ mogą one być stosowane zarówno w wersji lewostronnej (jako M_L) lub prawostronnej (jako M_R), operator M w przykładach poniżej, o ile nie zaznaczono inaczej, będzie mógł oznaczać zarówno M_L jak i M_R.

Ściskanie metodą Jacobiego Ściskanie metodą Jacobiego możemy zastosować tylko wtedy, gdy $a_{ii} \neq 0$ dla wszystkich i. Wtedy — przez analogię do metody iteracyjnej Jacobiego — wybieramy $M = \mathrm{diag}(A)^{-1}$. Oprócz zasadniczej wady, jaką jest najczęściej kiepska skuteczność w redukcji uwarunkowania, ściskanie metodą Jacobiego ma poza tym wiele zalet:
— generowanie M jest praktycznie darmowe,
— mnożenie przez M jest bardzo tanie ($O(N)$ flopów), a przy tym pięknie się zrównolegla.

Metody niepełnego rozkładu Inne sposoby ściśnięcia macierzy obejmują np. techniki tzw. *niepełnego rozkładu* macierzy. Ich idea opiera się na obserwacji, że „idealną" macierzą ściskającą byłaby $M = A^{-1}$. Przypomnijmy, że w takim przypadku mnożenie $M \cdot y$ najlepiej wykonać, gdy zawczasu wyznaczymy czynniki rozkładu $A = LU$ (co może nas drogo kosztować) i wtedy $My = U^{-1}(L^{-1}y)$ (tu wykorzystamy fakt, że układy równań z macierzami L i U są „łatwiejsze" do rozwiązania, bo macierze są trójkątne). Oczywiście, taka „idealna" macierz ściskająca ma wadę, która w większości przypadków będzie ją całkowicie dyskwalifikować: mimo, że A jest rozrzedzona, macierze L i U mogą — i zazwyczaj faktycznie są — gęste. Oznacza to, że samo wyznaczenie rozkładu LU macierzy A będzie nas kosztować zabójcze $O(N^3)$ flopów.

Inżynierski pomysł na usunięcie tej wady jest taki, by w algorytmie wyznaczania czynników rozkładu LU wyznaczać jedynie te elementy l_{ij} dolnego trójkąta L oraz u_{ij} górnego trójkąta U, dla których $a_{ij} \neq 0$. Zatem struktura niezerowych elementów macierzy L, U będzie powielać strukturę niezerowych elementów A — a więc dostaniemy w efekcie macierze rzadkie!

Jeśli więc ogólnie algorytm rozkładu LU (w miejscu, bez wyboru elementu głównego) jest postaci:

Listing.

for $k = 1$ **to** $N - 1$

```
  for i = k + 1 to N
    a_ik = a_ik/a_kk
  end
  for j = k + 1 to N
    for i = k + 1 to N
      a_ij = a_ij − a_ik ∗ a_kj
    end
  end
end
```

to niepełny rozkład LU miałby (formalnie) postać:

Niepełny rozkład LU, wersja bazowa

```
for k = 1 to N − 1
  for i = k + 1 to N
    if a_ik ≠ 0
      a_ik = a_ik/a_kk
    end
  end
  for j = k + 1 to N
    for i = k + 1 to N
      if a_ij ≠ 0
        a_ij = a_ij − a_ik ∗ a_kj
      end
    end
  end
end
```

Opisana powyżej technika nosi nazwę niepełnego rozkładu LU (ang. *incomplete LU factorization*) o zerowym wypełnieniu (ang. *zero fill-in*), co oznaczamy ILU(0). Warto zwrócić uwagę, że może zdarzyć się, że czynnik U uzyskany niepełnym rozkładem macierzy nieosobliwej sam będzie macierzą osobliwą. Dla pewnych klas macierzy — np. dla tzw. M-macierzy [13] — wiadomo, że niepełny rozkład nie prowadzi do takich degeneracji.

Często rozkład ILU(0) produkuje czynniki LU, dla których $U^{-1}L^{-1}$ jest tak bardzo odległe od A^{-1}, że nie dają one dostatecznie silnego ściśnięcia macierzy A. Wtedy można zastosować któryś z wariantów niepełnego rozkładu

— rozkład z progiem, ILUT (ang. *treshold ILU*), w którym zerujemy te elementy czynników rozkładu, które spełniają pewien warunek progowy — ogólnie mówiąc, warunek ten dotyczy względnej wielkości danego elementu („pomijamy małe”);
— rozkład według wzorca: w którym wprost wskazujemy, jakie elementy L i U mogą przyjąć niezerowe wartości;
— rozkład według wypełnienia, ILU(p): w którym dopuszczamy pewne dodatkowe wypełnienie czynników rozkładu.

Szczegółowe omówienie różnych technik niepełnego rozkładu macierzy, razem z ich analizą w niektórych praktycznie użytecznych przypadkach, znajdziemy w monografii Saada [13].

Metody dopasowane do problemu Najskuteczniejsze operatory ściskające dostajemy wykorzystując do maksimum całą specyfikę konkretnego rozwiązywanego zadania. Na przykład, dla macierzy pochodzących z dyskretyzacji niektórych równań różniczkowych cząstkowych opracowano kilka klas znakomitych sposobów ściskania takich, które gwarantują *bardzo szybką* zbieżność metod Kryłowa. Wśród nich znajdują się *metody dekompozycji obszaru* (por. rozdział 5.4, [14], [16], [**?**]) oraz *metody wielosiatkowe*. O tych ostatnich piszemy nieco więcej w rozdziale 8.3.

7.2. Macierze spektralnie równoważne

W wielu zastosowaniach, na przykład w wielokrotnie tu przywoływanych zadaniach dyskretyzacji eliptycznych równań różniczkowych cząstkowych, przedmiotem naszego zainteresowania jest nie tyle jeden układ równań, ale cała ich rodzina,

$$A_h x_h = b_h,$$

indeksowana pewnym parametrem h. Rzeczywiście, w przypadku zadań takich jak równanie z macierzą dyskretnego laplasjanu (zob. przykłady 5.1 i 5.2), naszym parametrem h byłby bok oczka siatki dyskretyzacji.

Powstaje więc naturalna potrzeba wskazania klasy macierzy ściskających, których użycie pozwalałoby uzyskać szybkość zbieżności metody iteracyjnej niezależną od h. W przypadku, gdy macierze A_h są symetryczne i dodatnio określone, macierzy ściskających warto szukać w tej samej klasie.

Definicja 7.1. Rodzinę macierzy $B_h = B_h^T > 0$ indeksowaną parametrem h nazywamy *spektralnie równoważną* rodzinie macierzy $A_h = A_h^T > 0$, jeśli istnieją dwie stałe c, C niezależne od h takie, że

$$c\, x^T B_h x \leqslant x^T A_h x \leqslant C\, x^T B_h x \quad \forall x \in \mathbb{R}^N. \tag{7.4}$$

Ważną własnością macierzy spektralnie równoważnych jest to, że na ich podstawie można skonstruować rodzinę macierzy ściskających, prowadzących do zadania o własnościach spektralnych niezależnych od h, por. [6].

Twierdzenie 7.1. *Jeśli B_h jest rodziną macierzy spektralnie równoważnych macierzy A_h, to wartości własne $\lambda(B_h^{-1}A_h)$ są ograniczone niezależnie od h.*

Dowód. Pokażemy nieco więcej: jeśli zachodzi warunek spektralnej równoważności (7.4), to dla dowolnego h

$$c \leqslant \lambda(B_h^{-1}A_h) \leqslant C.$$

Ponieważ dla dowolnej macierzy X, jej wartości własne są tożsame z wartościami macierzy $B_h^{1/2} X B_h^{-1/2}$, to w szczególności $B_h^{-1}A_h$ ma spektrum identyczne z macierzą $B_h^{-1/2} A_h B_h^{-1/2}$. Ta ostatnia jest symetryczna i dodatnio określona, więc wszystkie jej wartości własne są rzeczywiste i dodatnie.

Podstawiając $x = B^{-1/2}y$ w (7.4) i dzieląc stronami przez $y^T y$ dostajemy

$$c \leqslant \frac{y^T B_h^{-1/2} A_h B_h^{-1/2} y}{y^T y} \leqslant C \quad \forall y \in \mathbb{R}^N.$$

Znaczy to, że wartości własne macierzy $B_h^{-1/2} A_h B_h^{-1/2}$ leżą w przedziale $[c, C]$. □

Ćwiczenie 7.1. Wykaż, że jeśli zachodzi (7.4), to

$$c\lambda_{\min}(B_h^{-1}) \leqslant \lambda(A_h) \leqslant C\lambda_{\max}(B_h^{-1}) \quad \forall x \mathbb{R}^N.$$

(Por. [6, 3.10].)

Wskazówka. Wynika z twierdzenia Couranta–Fishera.

Ćwiczenie 7.2. Wykaż, że jeśli zachodzi (7.4), to jednocześnie zachodzi

$$\frac{1}{C}\, x^T B_h^{-1} x \leqslant x^T A_h^{-1} x \leqslant \frac{1}{c}\, x^T B_h^{-1} x \quad \forall x \mathbb{R}^N.$$

Rozwiązanie. Jak wiemy z dowodu twierdzenia 7.1, $\lambda(B_h^{-1/2} A_h B_h^{-1/2}) \subset [c, C]$. Ponieważ wartości własne macierzy odwrotnej są odwrotnościami wartości własnych macierzy danej, to w konsekwencji

$$\lambda(B_h^{1/2} A_h^{-1} B_h^{1/2}) \subset [1/C, 1/c].$$

Korzystając z faktu, że iloraz Rayleigh jest ograniczony z góry i z dołu przez ekstremalne wartości własne, dostajemy

$$\frac{1}{C} \leqslant \frac{y^T B_h^{1/2} A_h^{-1} B_h^{1/2} y}{y^T y} \leqslant \frac{1}{c} \quad \forall y \mathbb{R}^N.$$

Podstawiając $x = B_h^{1/2} y$ otrzymujemy tezę.

Przykład 7.1 (Wykorzystanie szybkiego *solvera* do ściśnięcia macierzy dyskretyzacji równania różniczkowego o zmiennych współczynnikach)**.** Rozważmy eliptyczne równanie różniczkowe

$$-\operatorname{div}(C(x)\nabla u(x)) = f(x), \qquad x \in \Omega = [0,1]^d, \tag{7.5}$$

$$u(x) = 0, \qquad x \in \partial\Omega, \tag{7.6}$$

gdzie $C(x)$ jest rzeczywistą i symetryczną macierzą rozmiaru $d \times d$, o współczynnikach będących ciągłymi, ograniczonymi funkcjami na Ω, spełniającą warunek (gwarantujący eliptyczność w/w równania różniczkowego)

$$\exists \gamma > 0 \quad \forall x \in \Omega \qquad v^T C(x) v \geqslant \gamma v^T v \qquad \forall v \in \mathbb{R}^d.$$

Stąd między innymi wynika, że forma dwuliniowa określona na przestrzeni Sobolewa $V = H_0^1(\Omega)$,

$$a(u,v) = \int_\Omega C(x)\nabla u(x) \cdot \nabla v(x)\, dx,$$

jest V–eliptyczna, to znaczy,

$$a(u,u) \geqslant \gamma \|u\|_V^2 \qquad \forall u \in V,$$

gdzie $\|u\|_V^2 = \int_\Omega \nabla u(x) \cdot \nabla v(x)\, dx$. Forma $a(\cdot,\cdot)$ jest także ciągła,

$$a(u,v) \leqslant \Gamma \|u\|_V \|u\|_V \qquad \forall u, v \in V.$$

Stałe γ, Γ zależą oczywiście od C.

Nasze zadanie w postaci wariacyjnej: znaleźć $u \in V$ takie, że

$$a(u,v) = f(v) \equiv \int_\Omega f(x)\, v(x)\, dx \qquad \forall v \in V$$

będziemy aproksymować metodą elementu skończonego w pewnej podprzestrzeni skończonego wymiaru $V^h \subset V$ (por. wykład z Numerycznych równań różniczkowych). Będziemy więc szukać $u^h \in V^h$ takiego, że

$$a(u^h, v^h) = f(v^h) \qquad \forall v^h \in V^h.$$

Reprezentując[2] u^h w bazie $\{\nu_1^h, \ldots, \nu_N^h\}$ przestrzeni V^h: $u^h = \sum_{i=1}^N U_i^h \nu_i$ mamy, że nasze zadanie dyskretne jest równoważne znalezieniu $U_h \in \mathbb{R}^N$ spełniającego układ równań

$$A_h U_h = F_h, \tag{7.7}$$

[2] Oczywiście, N zależy od h, ale nie będziemy tego wyraźnie zaznaczać, by nie mnożyć indeksów.

gdzie $(A_h)_{ij} = a(\nu_i^h, \nu_j^h)$. (Macierz A_h nazywana jest macierzą sztywności.) Gdy $d = 3$, metody bezpośrednie mogą być mało efektywne wskutek problemów z wypełnieniem czynników rozkładu, należy więc układ (7.7) rozwiązywać iteracyjnie. Niestety, w przypadku, gdy bazę wybierzemy jako standardową bazę elementu skończonego, macierz A_h jest bardzo źle uwarunkowana ze względu na h: $\mathrm{cond}_2(A_h) = O(h^{-2})$ i dlatego konieczne jest solidne jej ściśnięcie.

Niech B_h będzie macierzą sztywności uzyskaną w przypadku, gdy $C(x) = I$, czyli — „zwykłą" macierzą dyskretnego laplasjanu.

Zauważmy, że przy zachowaniu wyżej wspomnianej odpowiedniości pomiędzy funkcją $u_h \in V^h$ a jej reprezentacją $U_h \in \mathbb{R}^N$ w wybranej bazie zachodzi następujący związek pomiędzy macierzą sztywności, a formą dwuliniową:

$$U_h^T A_h U_h = a(u_h, u_h),$$

i w konsekwencji także

$$U_h^T B_h U_h = \|u_h\|_V^2.$$

Na mocy ciągłości i eliptyczności formy a w normie $\|\cdot\|_V$, macierze A_h i B_h są spektralnie równoważne.

A więc, znając szybką metodę rozwiązywania zadania z macierzą B_h — lub, ogólniej, dobrą macierz ściskającą dla B_h — dysponujemy jednocześnie dobrym (czytaj: niezależnym od h) sposobem ściśnięcia macierzy A_h.

Aby jednak trochę zbalansować nasz pogląd na tę sprawę warto zauważyć, że swój wkład w uwarunkowanie A_h ma także stosunek Γ/γ. Jeśli jest on bardzo duży (a może tak być, np. w materiałach silnie anizotropowych), wtedy uwarunkowanie $B_h^{-1}A_h$ — choć już nie będzie zależało od h — jednak wciąż będzie wielkie: nad tym, jak najlepiej ściskać takie macierze są obecnie prowadzone badania.

7.3. Metoda PCG

Naturalnym kandydatem na macierz ściskającą w metodzie CG (która jest określona dla macierzy A symetrycznej i dodatnio określonej) jest inna macierz symetryczna i dodatnio określona, M. Jak widzieliśmy w twierdzeniu 7.1 i ćwiczeniu 7.2, dobrym kandydatem na M mogłaby być macierz spektralnie równoważna macierzy A^{-1}. Wydawać by się jednak mogło, że jednostronne (na przykład, lewostronne) ściskanie taką macierzą nie powiedzie się, bo otrzymana przez nas macierz, MA, w ogólnosci nie musi być dalej symetryczna — wbrew założeniom uczynionym w wyprowadzeniu metody CG.

Na szczęście, macierz MA *jest* symetryczna, ale w *niestandardowym* iloczynie skalarnym: $\langle x, y\rangle = x^T M^{-1} y$. Aby uniknąć przeformułowania wyników uzyskanych we wcześniejszych rozdziałach, użyjemy pewnego triku: *teoretycznie* użyjemy zadania symetrycznie ściśniętego obustronnie macierzami $M^{1/2}$:

$$M^{1/2} A M^{1/2} \tilde{x} = M^{1/2} b, \tag{7.8}$$

$$x = M^{1/2} \tilde{x}, \tag{7.9}$$

ale *implementację* doprowadzimy do takiego stanu, że będzie w niej występować tylko mnożenie wektora przez macierz M — i to tylko jeden raz w każdej iteracji!

Rzeczywiście, ściśnięta macierz $\tilde{A} = M^{1/2} A M^{1/2}$ jest symetryczna i dodatnio określona, można więc do niej zastosować bezpośrednio algorytm CG, wyznaczający kolejne przybliżenia $\tilde{x}_k \in \tilde{x}_0 + K_k(\tilde{r}_0, \tilde{A})$. Oznaczając $\tilde{b} = M^{1/2} b$, na k-tej iteracji wyznaczymy:

— $\tilde{r}_k = \tilde{b} - \tilde{A}\tilde{x}_k$,

— $\tilde{p}_k = \tilde{r}_k + \beta\tilde{p}_{k-1}$
— $\rho_k = \|\tilde{r}_k\|_2^2$
— $\alpha = \rho_{k-1}/\tilde{p}_{k-1}^T\tilde{A}\tilde{p}_{k-1}$
— $\beta = \rho_k/\rho_{k-1}$
— $\tilde{x}_{k+1} = \tilde{x}_k + \alpha\tilde{p}_{k-1}$

Wprowadzając oznaczenia $r_k = b - Ax_k$, gdzie $x_k = M^{1/2}\tilde{x}_k$, i przyjmując $p_k = M^{1/2}\tilde{p}_k$ możemy zapisać powyższe operacje wyłącznie w terminach wartości „bezfalkowych”:

— $\tilde{r}_k = M^{1/2}(b - Ax_k) = M^{1/2}r_k$,
— $p_k = r_k + \beta p_{k-1}$ (wynika z pomnożenia odpowiedniej równości z falkami przez $M^{1/2}$)
— $\rho_k = \|\tilde{r}_k\|_2^2 = r_k^T M r_k$
— $\alpha = \rho_{k-1}/\left((M^{-1/2}p_{k-1})^T M^{1/2}AM^{1/2}M^{-1/2}p_{k-1}\right) = \rho_{k-1}/p_{k-1}^T A p_{k-1}$
— $\beta = \rho_k/\rho_{k-1}$
— $x_{k+1} = x_k + \alpha p_{k-1}$ (wynika z pomnożenia odpowiedniej równości z falkami przez $M^{1/2}$).

Znaczy to, że cały algorytm PCG możemy w praktyce zaimplementować tak, jak algorytm CG z tą różnicą, że do wyznaczenia ρ_k będziemy musieli obliczyć dodatkowy wektor Mr_k. Tym samym w implementacji nie pojawi się nigdzie mnożenie przez $M^{1/2}$ — w każdej iteracji będzie potrzebne tylko jedno mnożenie przez M!

Przykład 7.2. Działanie PCG z dobrą macierzą ściskającą prześledzimy na przykładzie macierzy jednowymiarowego laplasjanu, o losowo zaburzonych elementach:

$$A = T_N + S,$$

gdzie S jest symetryczną, trójdiagonalną macierzą o losowo wybranych elementach z przedziału $(0, \tau)$ (przy czym τ jest małe), a T_N zadano wzorem (5.5).

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, info, iter, resid] = jacobi(A, b, tol=1e-10, maxit=1000, x0)
%
% metoda Jacobiego na macierzy A
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch7.

Sprawdź, czy szybkość zbieżności PCG zależy od N i od τ. Zobacz, jak zmienią się rezultaty, gdy zaburzenie nie będzie zachowywać symetrii lub dodatniej określoności A. A co stanie się, gdy S nie będzie trójdiagonalna?

Ćwiczenie 7.3. Wykaż, że w metodzie PCG $x_k \in x_0 + K_k(Mr_0, MA)$.

Rozwiązanie. Ponieważ $\tilde{x}_k \in \tilde{x_0} + K_k(\tilde{r}_0, \tilde{A})$, to mnożąc stronami przez $M^{1/2}$ mamy, że $x_k \in x_0 + M^{1/2}K_k(\tilde{r}_0, \tilde{A})$. Ponieważ

$$\begin{aligned} M^{1/2}\tilde{A}^j\tilde{r}_0 &= M^{1/2}(M^{1/2}AM^{1/2})^j M^{1/2}r_0 \\ &= M^{1/2}\cdot M^{1/2}AM^{1/2}\cdot M^{1/2}AM^{1/2}\cdots M^{1/2}AM^{1/2}\cdot M^{1/2}r_0 \\ &= (MA)^j Mr_0, \end{aligned}$$

to

$$M^{1/2}K_k(\tilde{r}_0, \tilde{A}) = K_k(Mr_0, MA).$$

7.4. Ściskanie dla GMRES

Ponieważ metoda GMRES nie wymaga żadnych specjalnych własności macierzy (z wyjątkiem nieosobliwości), po obustronnym ściśnięciu[3] macierzy A nieosobliwymi macierzami M_L i M_R:

$$M_LAM_Ry = M_Lb$$

wystarczy do macierzy M_LAM_R zastosować standardowy algorytm GMRES i pamiętać, by wyznaczone rozwiązanie przybliżone y_k przetransformować do rozwiązania oryginalnego układu:

$$x_k = M_Ry_k.$$

Wszelkie mnożenia przez macierz ściśniętą będziemy oczywiście realizować zgodnie z nawiasami:

$$M_LAM_Rv = M_L(A(M_Rv)).$$

Warto pamiętać, GMRES będzie obliczał residua równe $r_k = M_L(b - Ax_k)$, zatem tylko gdy stosujemy prawostronne ściskanie, residua wyznaczone algorytmem GMRES będą identyczne z residuami układu wyjściowego (co, generalnie, samo w sobie niekoniecznie musi być wartością na której powinno nam zależeć).

7.5. Kryteria stopu metod iteracyjnych

Do tej pory zajmowaliśmy się samymi metodami przybliżonego rozwiązywania równań, zwracając uwagę na koszt jednej iteracji, sposób implementacji czy też szybkość zbieżności. Jednak każdą iterację musimy kiedyś w końcu przerwać; musimy więc zadać sobie pytanie: kiedy i na jakiej podstawie powinniśmy zakończyć działanie metody iteracyjnej?

7.5.1. Kryterium błędu

Idealnym kryterium stopu metody iteracyjnej byłaby redukcja *błędu* poniżej zadanego poziomu, mierzonego w zadanej przez użytkownika normie. Kryterium błędu zatrzymania metody iteracyjnej może więc mieć jedną z trzech postaci:

Bezwzględne

$$\|x_k - x^*\| \leqslant \epsilon_{\text{abs}},$$

[3] Jednostronne ściśnięcie dostaniemy, biorąc jedną z macierzy równą identyczności.

Względne

$$\|x_k - x^*\| \leqslant \epsilon_{\text{rel}} \|x_0 - x^*\|,$$

Mieszane

$$\|x_k - x^*\| \leqslant \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \|x_0 - x^*\|.$$

Sęk w tym, że zazwyczaj nie możemy wyznaczyć $\|x_k - x^*\|$ z prostego powodu: nie znamy rozwiązania dokładnego x^*! Z drugiej strony, zawsze dysponujemy przecież inną, *obliczalną* wprost wartością: residuum $r_k = b - Ax_k$.

7.5.2. Kryterium residualne

Ponieważ $r_k = 0$ odpowiada znalezieniu rozwiązania dokładnego, to kryterium stopu możemy oprzeć na redukcji *residuum* poniżej zadanego poziomu. Kryterium residualne zatrzymania metody iteracyjnej może więc mieć jedną z trzech postaci:

Bezwzględne

$$\|r_k\| \leqslant \epsilon_{\text{abs}},$$

Względne

$$\|r_k\| \leqslant \epsilon_{\text{rel}} \|r_0\|,$$

Mieszane

$$\|r_k\| \leqslant \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \|r_0\|.$$

Ćwiczenie 7.4. Podaj przykład takiej normy, w której możemy dokładnie wyznaczyć $\|x_k - x^*\|$ na podstawie r_k.

Rozwiązanie. Pamiętajmy, że zakładamy zawsze, że A jest nieosobliwa. Ponieważ $r_k = b - Ax_k = A(x^* - x_k)$, to

$$\|r_k\|_2^2 = (x^* - x_k)^T A^T A (x^* - x_k) = \|x_k - x^*\|_{A^T A}^2.$$

Zatem standardowa norma euklidesowa residuum odpowiada normie energetycznej błędu liczonej w normie indukowanej przez macierz symetryczną i dodatnio określoną $A^T A$.

Jednak zależność między normą residuum a *tą samą* normą błędu może nie być bardzo wyrazista: po raz kolejny może tu dać znać o sobie złe uwarunkowanie macierzy.

Stwierdzenie 7.1 (o oszacowaniu błędu przez residuum)**.** *Niech x^* będzie rozwiązaniem układu $Ax = b$ i niech $r_k = b - Ax_k$ dla zadanego x_k. Wtedy*

$$\frac{1}{\text{cond}(A)} \frac{\|r_k\|}{\|b\|} \leqslant \frac{\|x_k - x^*\|}{\|x^*\|} \leqslant \text{cond}(A) \frac{\|r_k\|}{\|b\|},$$

gdzie $\text{cond}(A) = \|A\| \cdot \|A^{-1}\|$.

Powyższe stwierdzenie należy interpretować tak, że jedynie w przypadku, gdy macierz A jest dobrze uwarunkowana w danej normie, mała norma residuum jest dobrym indykatorem małej (tej samej) normy błędu. Warto wiedzieć, że w metodzie CG można niskim kosztem ($O(k)$, gdzie k jest liczbą iteracji) wyznaczyć (coraz lepsze, wraz z postępem iteracji) dolne oszacowanie na spektralny współczynnik uwarunkowania $\kappa = \lambda_{\max}(A)/\lambda_{\min}(A) = \|A\|_2 \cdot \|A^{-1}\|_2$.

7.5.3. Kryterium postępu

W praktyce obliczeniowej stosuje się dwa dodatkowe kryteria stopu metody iteracyjnej

Stagnacji

$$\|x_{k+1} - x_k\| \leqslant \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \|x_k\|,$$

Cierpliwości

$$k > k_{\max}.$$

Rzeczywiście, musimy przecież określić moment, kiedy należałoby zakończyć obliczenia, jeśli do tej pory nie przyniosły sukcesu: zrobimy to więc po $k_{\max}$ iteracjach. Ponadto, jeżeli poprawka $\|x_{k+1} - x_k\|$ jest niewielka, to metoda być może na dobre ugrzęzła w jakimś przybliżeniu i wtedy, jeśli nadal jesteśmy daleko od rozwiązania, należałoby spróbować czegoś innego. To kryterium należy stosować z dużym wyczuciem: być może stagnacja metody była tylko chwilowa i po kilku następnych iteracjach zbieżność przyspieszy?

Ćwiczenie 7.5. Rozważmy stacjonarną metodę iteracyjną

$$x_{k+1} = Mx_k + c,$$

przy czym $\|M\| \leqslant \beta < 1$. Wykaż, że jeśli

$$\|x_k - x_{k-1}\| \leqslant \epsilon \frac{1-\beta}{\beta},$$

to

$$\|x_k - x^*\| \leqslant \epsilon.$$

Rozwiązanie. Ponieważ w metodzie stacjonarnej $x^* = Mx^* + c$, to

$$\begin{aligned}\|x_k - x^*\| = \|Mx_{k-1} + c - x^*\| = \|M(x_{k-1} - x^*)\| \leqslant \|M\|\|x_{k-1} - x^*\| \leqslant \beta(\|x_{k-1} - x_k\| + \|x_k - x^*\|) \\ \leqslant \epsilon(1-\beta) + \beta\|x_k - x^*\|.\end{aligned}$$

Zatem $\|x_k - x^*\|(1-\beta) \leqslant \epsilon(1-\beta)$, skąd teza.

8. Przegląd innych metod rozwiązywania wielkich układów równań liniowych

Na szczęście, nie ma jednej najlepszej metody rozwiązywania wielkich układów równań liniowych; czyni to oczywiście życie znacznie ciekawszym. Doprowadziło to do skonstruowania dużej liczby konkurencyjnych metod: zarówno bezpośrednich, jak i iteracyjnych. Celem nieniejszego rozdziału jest pobieżne zaznajomienie Czytelnika z niewielką ich reprezentacją. Dobór tej właściwej dla konkretnego zadania nie musi być oczywisty! Szerszy przegląd takich metod znajdziemy m.in. w monografii [13], [**?**].

8.1. Inne metody Kryłowa

Na początek, na podstawie [1] przedstawimy w pewien systematyczny sposób uogólnienie metody CG na szerszą klasę problemów. Następnie zrobimy krótki przegląd wybranych metod Kryłowa, *nie opartych* na zasadzie minimalizacji.

8.1.1. Uogólnienia metody CG

Wielką zaletą metody PCG jest to, że nie wymaga ona pamiętania całej bazy przestrzeni Kryłowa $K_k(Mr_0, MA) = \operatorname{span}\{Mr_0, MAMr_0, \ldots, (MA)^{k-1}Mr_0\}$, a iteracja ma prostą formułę $x_{k+1} = x_k + d_k$, gdzie $d_k \in K_k(Mr_0, MA)$ jest tak dobrany by minimalizować pewną normę błędu. Można więc pójść krok dalej i określić całą klasę metod o podobnej strukturze.

Przyjmijmy więc, że

— macierz A jest nieosobliwa (na tym poziomie ogólności nie zakładamy symetrii ani dodatniej określoności)

oraz są dane dwie dodatkowe macierze:

— nieosobliwa macierz lewostronnie ściskająca M,

— symetryczna, dodatnio określona macierz B, która będzie definiować normę, w której będziemy minimalizować błąd:

$$\|x\|_B = (x^T Bx)^{1/2}.$$

Będziemy zakładali, że macierz ściśnięta MA jest normalna względem iloczynu skalarnego indukowanego przez macierz B, co możemy zapisać w formie warunku

$$GG^T = G^T G,$$

gdzie $G = B^{1/2}MAB^{-1/2}$.

W uogólnionej metodzie CG, którą będziemy oznaczali GCG(B, M, A), kolejna iteracja $x_{k+1} \in x_0 + K_k(Mr_0, MA)$ będzie określona tak, by

$$\|x_{k+1} - x^*\|_B \leqslant \|x - x^*\|_B \qquad \forall x \in x_0 + K_k(Mr_0, MA) \tag{8.1}$$

Ćwiczenia testowe 8.1. 1. Czy metoda CG to nic innego jak GCG(I, I, A)?
NIE. Komentarz do odpowiedzi prawidłowej: *Oczywiście, CG to nic innego jak GCG(A, I, A).* Komentarz do odpowiedzi błędnej: *Sprawdź, w jakiej normie CG minimalizuje błąd.*

2. Czy metoda PCG z macierzą ściskającą M to nic innego jak GCG(A, M, A)?
TAK. Komentarz do odpowiedzi prawidłowej: *Oczywiście*. Komentarz do odpowiedzi błędnej: *Zob. [1]*.

Twierdzenie 8.1. *Przy powyższych założeniach, metoda GCG(B, M, A) jest dobrze określona i w dokładnej arytmetyce osiąga rozwiązanie dokładne po co najwyżej N krokach.*

Dowód. Kładąc $\tilde{r}_0 = Mr_0$ oraz $\tilde{A} = MA$, możemy zastosować twierdzenie 6.1 do przestrzeni Kryłowa $K_k = K_k(\tilde{r}_0, \tilde{A})$, które zagwarantuje nam spełnienie tezy twierdzenia. □

Twierdzenie 8.2. *Przy powyższych założeniach, błąd na k-tym kroku metody GCG(B, M, A) spełnia oszacowanie*

$$\|x_k - x^*\|_B \leqslant \min_{p \in \tilde{P}_k} \max_{\lambda \in \sigma(MA)} |p(\lambda)| \|x_0 - x^*\|_B.$$

Dowód. Na mocy wniosku 6.1,

$$\|x_k - x^*\|_B = \min_{p \in \tilde{P}_k} \|p(MA)(x_0 - x^*)\|_B.$$

Ponieważ dla dowolnego y

$$\|(MA)^k y\|_B = \|(B^{1/2} MAB^{-1/2})^k \, B^{1/2} y\|_2, \tag{8.2}$$

to w konsekwencji $\|p(MA)y\|_B = \|p(B^{1/2}MAB^{-1/2})\, B^{1/2}y\|_2 \leqslant \|p(B^{1/2}MAB^{-1/2})\|_2 \, \|y\|_B$. Z założenia macierz $G = B^{1/2}MAB^{-1/2}$ jest normalna, jest więc diagonalizowalna i jej wektory własne są ortogonalne:

$$G = X\Lambda X^T, \qquad \text{gdzie } X^T X = I.$$

Stąd zaś wynika, że $\|p(G)\|_2 = \|Xp(\Lambda)X^T\|_2 = \|p(\Lambda)\|_2 = \max_{\lambda \in \sigma(G)} |p(\lambda)|$, co kończy dowód, gdyż macierze G i MA są podobne, więc mają to samo spektrum. □

Ćwiczenie 8.2. Udowodnij równość (8.2).

Można, postępując analogicznie jak w przypadku klasycznej metody CG wykazać, że GCG(B, M, A) ma własności podobne jak jej protoplastka:

Stwierdzenie 8.1. *W metodzie GCG(B, M, A) zachodzi:*
— $x_k = x_{k-1} + \alpha_k p_k$, *gdzie* p_k *tworzą bazę* B*-ortogonalną przestrzeni Kryłowa* $K_k(Mr_0, MA)$.
— $w^T B(x_k - x^*) = 0$ *dla każdego* $w \in K_k(Mr_0, MA)$.

Dowód. W oczywisty sposób $x_k = x_{k-1} + d_k$, gdzie $d_k \in K_k(Mr_0, MA)$. Jeśli przez V_k oznaczyć bazę w $K_k(Mr_0, MA)$, to $x_k = x_0 + V_k a_k$ i na mocy (6.2) a_k spełnia układ równań normalnych,

$$V_k^T B(V_k a_k - (x^* - x_0)) = 0.$$

Ponieważ $V_k a_k = x_k - x_0$, dostajemy

$$V_k^T B(x_k - x^*) = 0,$$

co dowodzi drugiego punktu.

□

Metoda	B	M	Ograniczenia
CG	A	I	$A = A^T > 0$
CR	A^2	I	$A = A^T$
PCG	A	M	$A = A^T > 0$, $M = M^T$
PCR	AMA	M	$A = A^T$, $M = M^T > 0$
CGNR	A^TA	A^T	
CGNE	I	A^T	
D'yakonov	B	$B^{-1}A^TB^{-1}$	$B = B^T > 0$

Tabela 8.1. Wybrane metody Kryłowa dające się zinterpretować jako metoda GCG. W tabeli zamieszczono założenia na macierze A i M gwarantujące poprawność określenia metody.

Metodę GCG(B, M, A) możnaby zrealizować następującym uogólnieniem algorytmu CG, który nazwiemy za [1] algorytmem Odir(B, M, A):

Metoda Odir dla GCG, wersja bazowa

$p_0 = Mr_0$; $k = 0$;
while not stop **begin**
$\alpha_k = \dfrac{p_k^T B(x^* - x)}{p_k^T B p_k}$
$x_k = x_{k-1} + \alpha_k p_k$
$r_k = r_{k-1} - \alpha_k A p_k$
$\gamma_k = \dfrac{p_k^T BMAp_k}{p_k^T Bp_k}$
$\sigma_k = \dfrac{p_k^T BMAp_{k-1}}{p_{k-1}^T Bp_{k-1}}$
$p_{k+1} = MAp_k - \gamma_k p_k - \sigma_k p_{k-1}$
$k = k + 1$
end

Algorytm ten jeszcze nie jest gotowy do użycia, albowiem wymaga obliczania $B(x^* - x)$ — a więc potencjalnie wymaga odwołania się do nieznanego *a priori* wektora błędu! Możliwość skutecznego *obliczania* współczynnika α_k ogranicza więc zbiór B, których możemy użyć do zdefiniowania konkretnej metody. Z drugiej strony, wybierając konkretne B możemy dalej uprościć powyższą bazową implementację. Na tym etapie nie jest także oczywiste, czy może zdarzyć się $p_k = 0$ — a więc stagnacja i załamanie się algorytmu (dzielenie przez zero!).

Wśród metod, które dają się skutecznie zaimplementować na podstawie algorytmu Odir(B, M, A) — to znaczy: mają *obliczalne* α_k — znajdują się metody wymienione w tabeli 8.1, którą przytaczamy za [1, Table 5.2]. Z tabeli wynika m.in., że PCG wcale nie wymaga *dodatnio* określonej macierzy ściskającej!

Aby otrzymać dobrą implementację konkretnej metody opartej na algorytmie Odir, należy zawsze ją dopracować, wykorzystując zależności specyficzne dla konkretnej metody.

Warta uwagi jest metoda PCR, która działa w przypadku, gdy macierz A jest jedynie symetryczna — nie musi być dodatnio określona. Metoda D'aykonova [6] co prawda działa dla dowolnej nieosobliwej, niesymetrycznej macierzy A, ale ze względu na jej podobieństwo do równań normalnych dla $B^{-1}A$, w wielu wypadkach skuteczniejsza (pod względem szybkości zbieżności, a nie pod względem oszczędności pamięci) będzie metoda typu GMRES.

Uwaga 8.1. Czasami wymaganie, by można było wykonywać mnożenie przez A^T może być trudne do spełnienia — na przykład wtedy, gdy A jest zadana jako operator, tzn. wyłącznie przez *procedurę* mnożenia przez zadany wektor x, tzn. obliczania $A \cdot x$.

Ćwiczenie 8.3. Porównaj zbieżność metod: PCG i PCR w przypadku, gdy A oraz M są macierzami symetrycznymi i dodatnio określonymi. Przeprowadź weryfikację eksperymentalną swoich przypuszczeń.

Wskazówka. W MATLABie dyponujesz gotową implementacją zarówno metody `pcg`*, jak i* `pcr`*.*

8.1.2. Metody nie oparte na minimalizacji

Zamiast określać kolejną iterację metody, jak dotychczas, przez pewien warunek minimalizacji, możemy inaczej położyć warunek na x_k. Na przykład, możemy wymagać by oprócz $x_k \in x_0 + K_k(r_0, A)$ zachodził warunek ortogonalności residuów:

$$v^T r_k = 0 \qquad \forall v \in K_k(r_0, A^T).$$

(Zwróć uwagę na to, że residua mają być ortogonalne nie do $K_k(r_0, A)$, tylko do $K_k(r_0, A^T)$.)

Tego typu warunek prowadzi m.in. do metody BiCG (ang. *Bi-Conjugate Gradient*) która wymaga mnożenia przez A^T i nie jest też zbyt stabilna. Jej modyfikacje: CGS i BiCG-stab — nie wymagają już mnożenia przez A^T, BiCG-stab jest też bardziej stabilna. Niestety, jak na razie dla BiCG-stab nie ma pełnej i satysfakcjonującej teorii zbieżności.

W przeciwieństwie do metod minimalizacji, kolejna iteracja może nie być realizowalna mimo, że nie osiągnięto jeszcze rozwiązania — mówimy wtedy potocznie o *załamaniu się* metody (ang. *breakdown*).

W innej metodzie, QMR (ang. *Quasi-Minimal Residual*) — i jej wariancie bez mnożenia przez macierz transponowaną: TFQMR (ang. *Transpose-Free QMR*) — kolejną iterację wybiera się tak, by zminimalizować (stosunkowo łatwą do obliczenia) wielkość, która ma tylko trochę wspólnego z normą residuum.

Można pokazać [9], że metoda TFQMR dla macierzy $N \times N$ (realizowana w dokładnej arytmetyce) w ciągu $\lceil (N+1)/2 \rceil$ iteracji albo załamie się, albo osiągnie dokładne rozwiązanie.

8.2. Metody strukturalne

W bardzo wielu przypadkach, w macierzach układów równań jakie przychodzi nam rozwiązywać w praktycznych zastosowaniach, nie tylko niewiadome są powiązane jedynie z nielicznymi innymi niewiadomymi (co właśnie prowadzi do rozrzedzenia macierzy) ale w dość naturalny sposób można wskazać grupy niewiadomych i równań, które między sobą nie mają żadnych powiązań.

W najogólniejszym przypadku, mielibyśmy więc

$$A = \begin{pmatrix} A_{11} & & A_{13} \\ & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33}, \end{pmatrix} \tag{8.3}$$

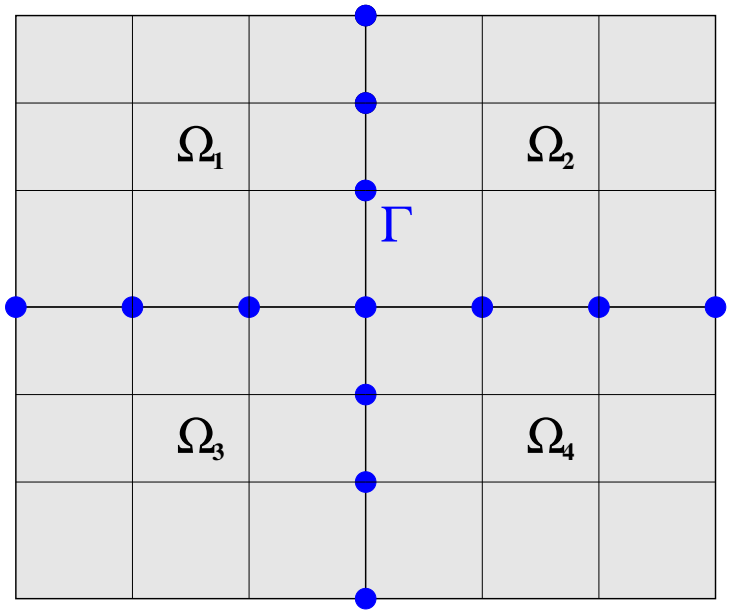
gdzie A_{ii} są pewnymi macierzami kwadratowymi. Niewiadome i równania z grupy „1” nie mają bezpośredniego związku z grupą „2” (są one powiązane ze sobą jedynie poprzez równania i niewiadome grupy „3”). W dalszym ciągu będziemy zakładać, że macierze A_{11} i A_{22} są nieosobliwe. *Strukturalizacja* równania liniowego to nic innego jak wskazanie pewnej permutacji

równań i niewiadomych, w wyniku której dostajemy układ postaci (8.3). W rzeczywistości, dla macierzy A_{11} i A_{22} mogłyby zachodzić podobne relacje[1], jak w poniższym przykładzie.

Przykład 8.1 (Strukturalizacja dyskretyzacji równania różniczkowego)**.** Rozważmy dyskretyzację równania różniczkowego

$$-\Delta u = f \text{ w } \Omega$$

z jednorodnymi warunkami brzegowymi Dirichleta. Jeśli obszar Ω podzielić na kilka mniejszych części, podobszarów, $\bar{\Omega} = \cup_{i=1}^{K} \bar{\Omega}_i$ i przez Γ oznaczyć zestaw krawędzi (ang. *interface*) je spajających: $\Gamma = \left(\cup_{i=1}^{K} \partial\Omega_i\right) - \partial\Omega$, to — ze względu na lokalny charakter działania pochodnej (i jej sensownych aproksymacji) — niewiadome z dwóch różnych podobszarów Ω_i i Ω_j nie będą ze sobą powiązane. Będą za to powiązane z niewiadomymi leżącymi na krawędziach spajających Γ, por. rysunek 8.1.



Rysunek 8.1. Przykładowy podział obszaru na $K = 4$ podobszary; węzły wspólne (należące do Γ) zaznaczono kolorem niebieskim.

Numerując niewiadome tak, by $U = (U_1, U_2, \ldots, U_K, U_\Gamma)^T$, gdzie wektor U_i zawiera niewiadome odpowiadające wewnętrznym węzłom w Ω_i, a U_Γ — niewiadome odpowiadające węzłom na interfejsie Γ, otrzymalibyśmy macierz o blokowej strukturze

$$D_h = \begin{pmatrix} D_{11} & & & & D_{1\Gamma} \\ & D_{22} & & & D_{2\Gamma} \\ & & \ddots & & \vdots \\ & & & D_{KK} & D_{K\Gamma} \\ D_{\Gamma 1} & D_{\Gamma 2} & \cdots & D_{\Gamma K} & D_{\Gamma\Gamma} \end{pmatrix}_{N\times N} . \tag{8.4}$$

W pustych miejscach D_h znajdują się bloki zerowe, a macierze D_{ii} faktycznie są nieosobliwe.

Dzięki szczególnej postaci macierzy (8.3) możemy podać algorytm typu „dziel i rządź” rozwiązywania układu równań $Ax = b$ z tą macierzą. Wprowadzając podział niewiadomych i

[1] Taką permutację nazywamy wtedy zagnieżdżonym podziałem grafu macierzy (ang. *nested dissection*).

wektora prawej strony zgodnie z blokowym podziałem macierzy A zadanym (8.3) mamy do rozwiązania układ

$$\begin{pmatrix} A_{11} & & A_{13} \\ & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

Dokonując blokowej eliminacji Gaussa (najpierw na blokach A_{11} i A_{22}) dostajemy następujący algorytm:

1. Oblicz $q_i = A_{ii}^{-1} b_i$, $i = 1, 2$.
2. Wyznacz x_3 jako rozwiązanie układu

$$Sx_3 = f,$$

   gdzie $S = A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23}$ oraz $f = b_3 - A_{31}q_1 - A_{32}q_2$.
3. Wyznacz $x_i = A_{ii}^{-1}(b_i - A_{i3}x_3)$, $i = 1, 2$.

Oczywiście, wszędzie tam, gdzie występują odwrotności macierzy, w praktyce będziemy rozwiązywali układy równań z tymi macierzami.

Zwróćmy uwagę na kilka charakterystycznych cech tego algorytmu:

— Zarówno w kroku 1. jak i w kroku 3. algorytmu, oba układy możemy rozwiązywać niezależnie od siebie (co daje możliwość równoległej implementacji).
— Macierze A_{ij} są rozrzedzone (bo A taka była), ale w ogólności S jest macierzą gęstszą.

Jeśli wymiar macierzy S jest niezbyt wielki, możemy ją wyznaczyć *explicite* i rozwiązać metodą bezpośrednią. Jednak jeśli S wciąż jest zbyt wielka (lub zbyt gęsta), by potraktować ją metodą bezpośrednią — wtedy możemy zastosować do niej metodę iteracyjną operatorową (*matrix-free*). Rzeczywiście, mnożenie macierzy S przez wektor z można zrealizować *bez* wyznaczania niej samej:

$$Sz = A_{33}z - A_{31}(A_{11}^{-1}(A_{13}z)) - A_{32}(A_{22}^{-1}(A_{23}z)).$$

(Nawiasy wskazują kolejność mnożenia.) Możemy więc wyznaczyć iloczyn Sz kosztem rozwiązania układów równań z macierzami A_{11} i A_{22} (co *i tak* musielibyśmy umieć zrobić!) oraz niewielkim dodatkowym kosztem pomnożenia pewnych wektorów przez rozrzedzone macierze trzeciej kolumny i trzeciego wiersza A.

Ćwiczenie 8.4. Przedyskutuj, jak rozwiązywać układ równań postaci (8.4).

8.3. Metody wielosiatkowe

Macierze rozrzedzone, które otrzymuje się z dyskretyzacji równań różniczkowych cząstkowych metodą różnic skończonych lub elementu skończonego, mają wiele specjalnych cech, które pozwalają na konstrukcję optymalnych metod rozwiązywania takich zadań. Jedną z takich klas metod są metody wielosiatkowe (ang. *multigrid*).

Omówimy jej ideę na przykładzie macierzy T_N dyskretyzacji jednowymiarowego laplasjanu (5.5): wszystkie istotne wady metod iteracyjnych ujawniają się już w jej przypadku, podobnie jak zasadnicze cechy metody wielosiatkowej. Oczywiście, w praktyce metodę wielosiatkową stosowalibyśmy dopiero do dyskretyzacji dwu- lub trójwymiarowego laplasjanu.

Gdyby przyjrzeć się metodzie Jacobiego dla T_N zobaczymy, że niektóre składowe błędu są w niej szybciej redukowane niż inne. Dokładniej, rozważmy metodę Jacobiego z parametrem relaksacyjnym ω,

$$x_{k+1} = x_k + \frac{\omega}{2} r_k$$

(w macierzy T_N diagonala jest stała równa $2I$). Macierz iteracji tej metody ma postać

$$G = I - \frac{\omega}{2} T_N.$$

Wyrażając błąd $e_k = x_k - x^*$ w bazie wektorów własnych macierzy G (są dane wzorem $(v_i)_j = \sin(\frac{ij\pi}{N+1})$ — por. (5.18)) dostajemy, że

$$e_0 = \sum_i a_i v_i$$

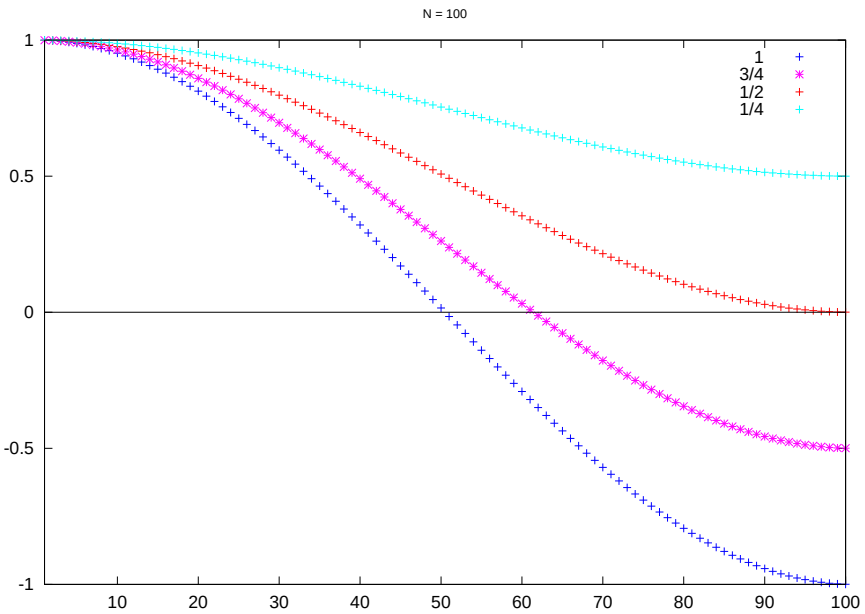
i w konsekwencji,

$$e_k = G^k e_0 = \sum_i \lambda_i(G)^k a_i v_i.$$

Jasne jest więc, że najszybszej redukcji ulegną te składowe błędu, które odpowiadają najbliższym zera wartościom $\lambda_i(G)$, najwolniej zaś będą znikać te, dla których $|\lambda_i(G)|$ będzie duży. Rysunek 8.2 przedstawia wykres

$$\lambda_i(G) = 1 - \omega(1 - \cos(\frac{i\pi}{N+1}))$$

w zależności od parametru relaksacyjnego ω.



Rysunek 8.2. Wartości własne macierzy iteracji metody Jacobiego z parametrem $\omega \in \{1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}\}$, zastosowanej do macierzy T_{100}.

Jak możemy zauważyć, gdy $\omega = 1$ (tzn. gdy używamy standardowej metody Jacobiego), najsłabiej są redukowane składowe szybkozmienne i wolnozmienne (jedynie te „średniozmienne" są redukowane błyskawicznie).

Ale już dla $\omega = 1/2$, najmocniej są redukowane wszystkie szybkozmienne składowe błędu — zatem po kilku iteracjach takiej metody błąd zostanie *wygładzony*. Jeśliby więc wspomóc taką prostą iterację poprawką, która redukowałaby pozostałe — wolnozmienne — składowe błędu, moglibyśmy otrzymać szybko zbieżną iterację. Jak pamiętamy z wcześniejszego wykładu, metody projekcji zastępowały idealną poprawkę $A\delta^* = r_k$ poprawką przybliżoną δ, wyznaczaną

przez rozwiązanie zadania zmniejszonego wymiaru. W metodzie *dwusiatkowej* poprawkę wyznaczymy podobnie: rozwiązując metodą bezpośrednią (np. eliminacji Gaussa) równanie poprawki na rzadszej siatce — wszak będzie ono tylko gorszą aproksymacją (o mniejszej rozdzielczości!) tego samego rozwiązania równania Poissona, które ma aproksymować x^*.

W metodzie *wielosiatkowej* zadanie na rzadszej siatce (potrzebne do wyznaczenia poprawki na siatce najdrobniejszej) potraktujemy rekurencyjnie w ten sam sposób, otrzymując sekwencję zadań coraz mniejszego rozmiaru, którą w pewnym momencie oczywiście będziemy musieli przerwać i rozwiązać zadanie bardzo małego wyniaru metodą bezpośrednią.

Oto więc kolejne etapy jednej iteracji metody wielosiatkowej, wyznaczającej kolejne przybliżenie x_{k+1} na podstawie x_k:

1. Wygładzenie (ang. *pre-smoothing*) — zaczynając od x_k, wykonujemy kilka iteracji (zwykle jedną lub dwie) prostej metody typu Jacobiego lub Gaussa–Seidela z właściwie dobranym parametrem ω. Dostajemy przybliżenie pośrednie $x^{(1)}_{k+1}$, ze zredukowaną szybkozmienną składową błędu.
2. Restrykcja — sformułowanie na rzadszej siatce zadania na poprawkę δ_c dla $x^{(1)}_{k+1}$.
3. Gruba poprawka (ang. *coarse grid correction*) — wyznaczenie poprawki δ_c na rzadszej siatce — przez iterację taką, jak ta! (Jako przybliżenie początkowe dla δ_c sensownie wziąć zero.)
4. Przedłużenie — wyrażenie poprawki obliczonej na rzadszej siatce w terminach wartości na gęstej siatki. Uwzględnienie poprawki i aktualizacja przybliżenia: $x^{(2)}_{k+1} = x^{(1)}_{k+1} + \delta$.
5. Dodatkowe wygładzenie (ang. *post-smoothing*) — ewentualnie. Znów kilka iteracji jak na pierwszym etapie, z przybliżeniem początkowym równym $x^{(2)}_{k+1}$, zakończone wyznaczeniem x_{k+1}. Gdy ten etap jest pominięty, za x_{k+1} bierze się $x^{(2)}_{k+1}$.

Okazuje się, że w ten sposób uzyskujemy metodę, której szybkość zbieżności na modelowym zadaniu nie zależy od parametru dyskretyzacji h (a więc i od rozmiaru zadania). Przypomnijmy, że było to zmorą wszystkich dotychczas rozważanych metod iteracyjnych; aby to przezwyciężyć, musieliśmy tam użyć spektralnie równoważnej macierzy ściskającej. Metoda wielosiatkowa nie potrzebuje tego, co więcej, można pokazać, że koszt jednej iteracji metody jest rzędu $O(N)$ — a więc (z dokładnością do stałej...) taki sam, jak koszt np. jednej iteracji metody CG.

Pełna metoda wielosiatkowa Idąc dalej tym tropem możemy zauważyć, że jako dobre przybliżenie początkowe dla (iteracyjnej) metody wielosiatkowej można byłoby wybrać przybliżone rozwiązanie na rzadszej siatce: wtedy jedynymi istotnymi składowymi błędu na gęstej siatce byłyby składowe szybkozmienne — a te wyeliminowalibyśmy w kilku iteracjach! Oczywiście, wymagane przybliżenie na rzadszej siatce uzyskalibyśmy w ten sam sposób z jeszcze rzadszej siatki; na siatce o największych oczkach moglibyśmy użyć jakiejś metody bezpośredniej.

Pamiętając, że rozwiązywane przez nas zagadnienie — układ równań liniowych — pochodzi z dyskretyzacji pewnego równania różniczkowego musimy zdać sobie sprawę z tego, że nie ma sensu rozwiązywać go z dokładnością większą niż dokładność aproksymacji samej dyskretyzacji! Dlatego iterację można przerwać po osiągnięciu z góry zadanego poziomu błędu aproksymacji rozwiązania zadania różniczkowego. Gdy będziemy startować z przybliżenia wyznaczonego metodą opisaną powyżej, okazuje się, że często wystarczy tylko niewielka, niezależna od h, liczba iteracji by osiągnąć zadany poziom błędu.

Taka metoda wyznaczania rozwiązania zadania powstałego po dyskretyzacji równania różniczkowego — którą tutaj tylko zasygnalizowaliśmy w uproszczeniu — nazywa się *pełną metodą wielosiatkową* (ang. *full multigrid*). Choć wciąż jest to metoda iteracyjna, ma ona cechy metody bezpośredniej. Okazuje się, że koszt wyznaczenia rozwiązania tą metodą jest rzędu $O(N)$ (kilku iteracji metody wielosiatkowej na finalnej, najdrobniejszej siatce) — a więc optymalny co do rzędu, gdyż samych niewiadomych w zadaniu jest N.

9. Układy równań nieliniowych. Metoda Newtona

Zajmiemy się teraz nową, trudniejszą klasą zadań: metodami rozwiązywania układów równań *nieliniowych*

$$F(x) = 0,$$

gdzie $F : \mathbb{R}^N \supset D \to \mathbb{R}^N$.

W matematyce stosowanej bardzo wiele modeli zyskuje na realizmie, gdy uwzględnić w nich zjawiska nieliniowe. Takie modele są jednak znacznie trudniejsze — zarówno w analizie, jak i w komputerowych symulacjach. Nieustanny wzrost mocy obliczeniowej komputerów, jaki obserwujemy w ciągu ostatnich kilkudziesięciu lat powoduje, że rozwiązywanie dużych układów nawet silnie nieliniowych równań stało się możliwe nawet dla posiadaczy „zwykłych" komputerów osobistych.

Od razu zauważmy, że już na poziomie matematycznej poprawności tego zadania możemy napotkać kłopoty, albowiem to zadanie może nie mieć rozwiązań, może mieć dokładnie jedno rozwiązanie, albo skończenie wiele rozwiązań, albo nieskończenie wiele izolowanych rozwiązań, albo... i tak dalej, i tak dalej.

Aby więc szukać jakiegokolwiek rozwiązania powyższego równania, musimy najpierw mieć pewność, że ono istnieje. Nie będziemy się tą kwestią tu zajmować, traktując ją jako domenę matematyki *czystej*, bez numerycznego zacięcia.

Z kursu *Matematyki obliczeniowej I* znamy kilka metod rozwiązywania nieliniowego równania skalarnego, $f(x) = 0$: metodę bisekcji, stycznych, siecznych, ... Okazuje się, że niektóre z nich da się w mniej lub bardziej naturalny sposób uogólnić na przypadek wielowymiarowy. W niniejszym rozdziale wyjaśnimy, jak metodę stycznych (Newtona) uogólnić na przypadek wielowymiarowy.

9.1. Szybkość zbieżności

Rozważmy ciąg $(x_k) \subset \mathbb{R}^N$ taki, że $x_k \to x^* \in \mathbb{R}^N$ gdy $k \to \infty$.

Definicja 9.1 (Zbieżność wykładnicza)**.** Powiemy, że $x_k \to x^*$ wykładniczo z rzędem co najmniej $q > 1$, jeśli

$$\exists C \geqslant 0 \; \exists K \in \mathbb{N} \quad \|x_{k+1} - x^*\| \leqslant C\|x_k - x^*\|^q \qquad \forall k \geqslant K. \tag{9.1}$$

Jeśli wykładnik zbieżności $q = 2$, mówimy wtedy o zbieżności kwadratowej; gdy $q = 3$, zbieżność nazywamy sześcienną (albo: kubiczną). W praktyce, zbieżność kwadratową uznaje się za satysfakcjonująco szybką, a zbieżność kubiczną — za bardzo szybką. Jednak nie zawsze taką szybkość zbieżności można uzyskać; rozsądnym praktycznym minimum jest zbieżność liniowa, w której błąd w każdym kroku maleje o stały czynnik:

Definicja 9.2 (Zbieżność liniowa)**.** Powiemy, że $x_k \to x^*$ przynajmniej liniowo, jeśli

$$\exists 0 \leqslant \alpha \leqslant 1 \; \exists K \in \mathbb{N} \quad \|x_{k+1} - x^*\| \leqslant \alpha\|x_k - x^*\| \qquad \forall k \geqslant K.$$

Jednak gdy $\alpha \approx 1$, zbieżność liniowa może być w praktyce nieakceptowalnie wolna.

Oczywiście powyższe nie wyczerpuje wszystkich możliwych szybkości zbieżności. Mianowicie, ciąg może być zbieżny szybciej niż liniowo (choć niekoniecznie zaraz kwadratowo):

Definicja 9.3 (Zbieżność superliniowa)**.** Powiemy, że $x_k \to x^*$ superliniowo, jeśli

$$\limsup_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Czasem trudno o samym ciągu powiedzieć, jaki ma wykładnik zbieżności, natomiast łatwiej jest wskazać ciąg o znanym wykładniku zbieżności, który majoryzuje błąd popełniany na każdej iteracji.

Definicja 9.4 (r–zbieżność)**.** Powiemy, że $x_k \to x^*$ r–wykładniczo/r–liniowo/r–superliniowo, jeśli istnieje ciąg $(\xi_k) \subset \mathbb{R}$ taki, że

$$\|x_k - x^*\| \leqslant \xi_k$$

oraz $\xi_k \to 0$, odpowiednio, wykładniczo/liniowo/superliniowo.

Aby móc porównywać (jakościowo) różne metody iteracyjne rozwiązywania równań nieliniowych — o różnych kosztach jednej iteracji i o różnych wykładnikach zbieżności — wprowadza się pojęcie wskaźnika efektywności metody.

Definicja 9.5 (Efektywność metody iteracyjnej)**.** Wskaźnik efektywności E metody iteracyjnej, zbieżnej wykładniczo z wykładnikiem q, której jedna iteracja kosztuje m flopów, definiujemy jako [17]

$$E = q^{1/m}.$$

Wskaźnik efektywności metody służy szybkiemu jakościowemu porównywaniu metod. Motywacją dla takiej definicji jest wymaganie, by przy zadanym (tym samym) całkowitym koszcie iteracji W obu metod, porównać uzyskane wykładniki redukcji błędu. Jeśli bowiem metoda jest rzędu $q > 1$, to błąd początkowy po k iteracjach zostanie zredukowany do poziomu $\|x_0 - x^*\|^{q^k}$. Ponieważ całkowity koszt k iteracji wynosi $W = km$, to $q^k = q^{W/m} = (q^{1/m})^W = E^W$. Definiując w ten sposób efektywność metody iteracyjnej, czynimy jednak pewne oproszczenie, polegające na tym, że pomijamy wpływ *stałych* na redukcję błędu (patrz oszacowanie (9.1)). Z tego powodu na przykład (formalnie) każda metoda zbieżna liniowo formalnie miałaby taką samą efektywność — równą 1 — choć oczywiście *realna* sprawność różnych metod liniowych może być różna, m.in. właśnie w zależności od stałej redukcji błędu.

Rozważmy dwa przykłady dużych układów równań nieliniowych, motywowanych — podobnie jak już kilka razy wcześniej w trakcie tego przedmiotu — zadaniami z równań różniczkowych cząstkowych.

Przykład 9.1 (stacjonarne równanie Allena–Cahna)**.** Rozważmy równanie

$$\begin{aligned} -\Delta u + \delta u(1-u^2) &= g(x,y) && \text{w } \Omega, \\ u &= 0 && \text{na } \partial\Omega. \end{aligned}$$

Dla uproszczenia przyjmiemy, że Ω jest kwadratem jednostkowym. Wprowadzając dyskretyzację tego równania najprostszą[1] metodą — metodą różnic skończonych na jednorodnej siatce — patrz

[1] Po raz kolejny przypomnijmy, że (z wyjątkiem trywialnych przypadków) najprostsze metody dyskretyzacji, choć bardzo ułatwiają programowanie, to jednak są zwykle najmniej efektywnym sposobem rozwiązywania równań różniczkowych, jeśli wziąć pod uwagę koszt obliczeniowy potrzebny do uzyskania satysfakcjonującego przybliżenia rozwiązania.

przykład 5.2 — otrzymujemy, po uwzględnieniu warunku brzegowego, nieliniowy układ równań algebraicznych,

$$P_N U_N + f(U_N) = 0,$$

gdzie P_N jest operatorem liniowym zdefiniowanym w przykładzie 5.2, natomiast dla zadanej macierzy $U \in \mathbb{R}^{N\times N}$

$$(f(U))_{ij} = \delta u_{ij}(1 - u_{ij}^2) - g_{ij}.$$

(Przypomnijmy, że dokonujemy tutaj utożsamienia macierzy $N \times N$ z wektorem długości N^2, ktorego elementami są kolejne wiersze macierzy.)

Przykład 9.2 (ewolucyjne równanie Allena–Cahna)**.** Rozważmy równanie

$$\begin{aligned} &u_t - \Delta u + \delta u(1 - u^2) = g(x, y) && \text{w } \Omega, \\ &u = 0 && \text{na } \partial\Omega. \end{aligned}$$

Dla uproszczenia przyjmiemy, że Ω jest kwadratem jednostkowym. Wprowadzając dyskretyzację tego równania najprostszą metodą — metodą różnic skończonych na jednorodnej siatce po zmiennej przestrzennej oraz niejawnym schematem Eulera po zmiennej czasowej (patrz wykład z Numerycznych równań różniczkowych, otrzymujemy, po uwzględnieniu warunku brzegowego, nieliniowy układ równań algebraicznych z niewiadomą U_N^k,

$$\frac{U_N^k - U_N^{k-1}}{\tau} + P_N U_N^k + f(U_N^k) = 0,$$

gdzie P_N jest operatorem liniowym zdefiniowanym w przykładzie 5.2, a f jest zadane jak w poprzednim przykładzie. (Przypomnijmy, że dokonujemy tutaj utożsamienia macierzy $N \times N$ z wektorem długości N^2, ktorego elementami są kolejne wiersze macierzy.) Zostawiając po lewej stronie tylko wyrazy z niewiadomymi, dostajemy równanie

$$(I + \tau P_N)\, U_N^k + \tau f(U_N^k) = U_N^{k-1},$$

a więc takie, w którym — dla dostatecznie małych τ — część nieliniowa staje się *mała* w porównaniu z U_N.

9.2. Metoda Banacha

Wiele równań nieliniowych można w naturalny sposób sformułować jako zagadnienie punktu stałego,

$$x = \Phi(x),$$

dla którego najprostszą metodą iteracyjną jest doskonale nam znana metoda Banacha (iteracji prostej),

$$x_{k+1} = \Phi(x_k).$$

Przypomnijmy (bez dowodu) twierdzenie o zbieżności tej metody:

Twierdzenie 9.1 (o zbieżności metody Banacha)**.** *Niech $D \subset \mathbb{R}^N$ będzie niepusty, ograniczony i domknięty, i niech $\Phi : D \to D$ będzie kontrakcją, tzn.*

$$\exists \gamma < 1 \quad \|\Phi(x) - \Phi(y)\| \leqslant \gamma \|x - y\| \qquad \forall x, y \in D.$$

Wtedy Φ ma dokładnie jeden punkt stały $x^* = \Phi(x^*) \in D$ oraz dla dowolnego $x_0 \in D$ iteracja

$$x_{k+1} = \Phi(x_k)$$

jest zbieżna do x^*, przy czym dla każdego $k = 0, 1, \ldots,$

$$\|x_{k+1} - x^*\| \leqslant \gamma \|x_k - x^*\|.$$

Z powyższego twierdzenia wynika więc, że iteracja Banacha jest zbieżna przynajmniej liniowo ze współczynnikiem redukcji błędu $\gamma < 1$.

Przykład 9.3. Stosując niejawny schemat Eulera[2]

$$y_{n+1} = y_n + \tau F(t_{n+1}, y_{n+1})$$

do przybliżonego rozwiązania układu równań różniczkowych zwyczajnych $y'(t) = F(t, y) \in \mathbb{R}^N$ musimy dla znanego y_n znaleźć y_{n+1} jako punkt stały odwzorowania

$$\Phi(x) = y_n + \tau F(t_{n+1}, x).$$

Jeśli F jest lipschitzowska ze stałą L w normie $\|\cdot\|$ ze względu na drugi argument (por. twierdzenie Picarda–Lindelöfa), to

$$\|\Phi(x) - \Phi(y)\| = \tau \|F(x) - F(y)\| \leqslant \tau L \|x - y\| \qquad \forall x, y \in \mathbb{R}^N,$$

zatem Φ jet kontrakcją na $\mathbb{R}^N$, gdy $\tau L < 1$, czyli gdy krok czasowy τ schematu Eulera jest dostatecznie mały względem h.

Czasami wygodniej jest skorzystać z „lokalnej" wersji twierdzenia o zbieżności metody Banacha:

Twierdzenie 9.2 (o lokalnej zbieżności metody Banacha)**.** *Niech $\Phi : \mathbb{R}^N \supset D \to \mathbb{R}^N$, gdzie D jest otwarty i niepusty, i niech dla zadanego $x_0 \in D$ istnieje kula $B = \{x \in \mathbb{R}^N : \|x - x_0\| \leqslant r\}$ taka, że $B \subset D$ oraz Φ jest na B kontrakcją ze stałą γ.*

Jeśli dodatkowo $\|x_0 - \Phi(x_0)\| \leqslant (1 - \gamma)r$, to w B istnieje dokładnie jeden punkt stały $x^* \in B$ odwzorowania Φ, a metoda iteracji prostej jest doń zbieżna przynajmniej liniowo ze stałą redukcji błędu γ.

Dowód. Wystarczy wykazać, że $\Phi(B) \subset B$ i skorzystać z poprzedniego twierdzenia. Na mocy warunku kontrakcji w B, dla $x \in B$ mamy

$$\|\Phi(x) - x_0\| \leqslant \|\Phi(x) - \Phi(x_0)\| + \|\Phi(x_0) - x_0\| \leqslant \gamma \|x - x_0\| + (1 - \gamma)r \leqslant \gamma r + (1 - \gamma)r = r,$$

zatem faktycznie $\Phi(x) \in B$. □

[2] Por. np. wykład z Numerycznych równań różniczkowych. Podobna metoda — pod nazwą metody łamanych Eulera — bywa wykorzystywana w dowodzie twierdzenia o istnieniu rozwiązania równania różniczkowego. Nas tutaj interesuje ona wyłącznie jako metoda numeryczna.

9.3. Metoda Newtona

W przypadku równania skalarnego, $f : \mathbb{R} \to \mathbb{R}$, metoda stycznych (zwana też metodą Newtona) rozwiązywania równania $f(x) = 0$ jest zadana wzorem

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Przez analogię, gdy $F : \mathbb{R}^N \supset D \to \mathbb{R}^N$ można więc byłoby zdefiniować *wielowymiarową metodę Newtona* wzorem

$$x_{k+1} = x_k - F'(x_k)^{-1}F(x_k), \tag{9.2}$$

gdzie $F'(x_k)$ oznaczałoby macierz pochodnej F w punkcie x_k. Jak za chwilę się przekonamy, jest to rzeczywiście bardzo dobry pomysł, a tak określona metoda zachowuje (z zachowaniem właściwych proporcji) wszystkie cechy metody Newtona znane nam z przypadku skalarnego!

9.3.1. Analiza zbieżności metody Newtona

Wielowymiarową metodę Newtona i różne jej warianty będziemy analizowali przy pewnych dość ogólnych założeniach dotyczących funkcji $F : \mathbb{R}^N \supset D \to \mathbb{R}^N$. W skrócie, będziemy je nazywali za [9] *założeniami standardowymi*:

1. Zbiór D jest otwarty i niepusty, a F jest różniczkowalna w D.
2. Istnieje rozwiązanie $x^* \in D$:
$$F(x^*) = 0.$$
3. Pochodna $F' : D \to L(\mathbb{R}^N)$ jest lipschitzowska ze stałą L:
$$\exists L > 0 \quad \|F'(x) - F'(y)\| \leqslant L\|x - y\| \qquad \forall x, y \in D,$$
przy czym norma po lewej stronie nierówności jest normą operatorową indukowaną przez normę wektorową w $\mathbb{R}^N$ obecną po prawej stronie, tzn. dla liniowego operatora $A : \mathbb{R}^N \to \mathbb{R}^N$,
$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$
4. Macierz pochodnej w rozwiązaniu, $F'(x^*)$, jest nieosobliwa.

Lematy techniczne

Przez $K(x^*, \delta)$ będziemy oznaczali kulę *otwartą* o środku w x^* i promieniu δ,

$$K(x^*, \delta) = \{x \in \mathbb{R}^N : \|x - x^*\| < \delta\}.$$

Lemat 9.1 (użyteczna wersja twierdzenia o wartości średniej)**.** *Niech będą spełnione założenia standardowe i niech K będzie otwartą kulą w D. Wtedy dla każdego $x, y \in K$,*

$$F(y) - F(x) = \int_0^1 F'\left(x + t(y - x)\right)(y - x)\, dt.$$

Dowód. Ustalmy $x, y \in K$. Ponieważ $K \subset D$ jest wypukły, to $x + t(y - x)$ dla $t \in [0, 1]$ i dla $t \in [0, 1]$ jest dobrze określona funkcja $g(t) = F(x + t(y - x))$.

Na mocy założeń standardowych $F'(\cdot)$ jest ciągła na $\bar{K}$, zatem także $g'(t) = F'(x + t(y - x))(y - x)$ jest ciągła na $[0, 1]$ (i całkowalna). Teza lematu wynika więc z podstawowego twierdzenia rachunku różniczkowego, że

$$\int_0^1 g'(t)\, dt = g(1) - g(0).$$

□

Lemat 9.2 (o lokalnym oszacowaniu funkcji i pochodnej)**.** *Przy założeniach standardowych, istnieje dostatecznie małe $\delta > 0$ takie, że dla dowolnego $x \in K(x^*, \delta)$ zachodzi:*

1. $\|F'(x)\| \leqslant 2\|F'(x^*)\|$,
2. $\|F'(x)^{-1}\| \leqslant 2\|F'(x^*)^{-1}\|$,
3. $\dfrac{1}{2\|F'(x^*)^{-1}\|}\|x - x^*\| \leqslant \|F(x)\| \leqslant 2\|F'(x^*)\|\|x - x^*\|$,

Ćwiczenie 9.1. Wykaż, że przy założeniach standardowych x^* musi być izolowanym rozwiązaniem równania $F(x) = 0$.

Rozwiązanie. Z lematu 9.2 o oszacowaniu funkcji, istnieje otoczenie $U \ni x^*$ takie, że

$$\frac{1}{2\|F'(x^*)^{-1}\|}\|x - x^*\| \leqslant \|F(x)\|.$$

Jeśli więc dla $\tilde{x}$ leżącego w tym otoczeniu $F(\tilde{x}) = 0$, to znaczyłoby to, że $\|\tilde{x} - x^*\| \leqslant 0$, a więc $\tilde{x} = x^*$. Czyli w U nie ma innych rozwiązań niż x^*.

Twierdzenie 9.3 (o zbieżności metody Newtona)**.** *Przy standardowych założeniach, istnieją $C > 0$ (dostatecznie duże) i $\delta > 0$ (dostatecznie małe) takie, że jeśli $x_0 \in K(x^*, \delta)$, to ciąg (x_k) zadany metodą Newtona (9.2) jest dobrze określony, $x_k \in K(x^*, \delta)$, oraz $x_k \to x^*$. Co więcej, ciąg ten jest zbieżny kwadratowo:*

$$\|x_{k+1} - x^*\| \leqslant C\|x_k - x^*\|^2 \qquad \forall k \in \mathbb{N}. \tag{9.3}$$

Dowód. Dowód będzie opierał się na wykazaniu oszacowania (9.3), pozostałe elementy tezy będą jego konsekwencją.

Na wstępie wybierzmy δ takie, by zachodził lemat 9.2 o oszacowaniu funkcji i pochodnej. Oznaczając $e_k = x_k - x^*$ mamy ze wzoru Newtona

$$e_{k+1} = e_k - F'(x_k)^{-1}F(x_k).$$

Na mocy założeń standardowych i lematu o wartości średniej,

$$F(x_k) = F(x_k) - F(x^*) = \int_0^1 F'(x^* + t(x_k - x^*))(x_k - x^*)\,dt = \int_0^1 F'(x^* + te_k)e_k\,dt,$$

zatem

$$e_{k+1} = e_k - F'(x_k)^{-1}\int_0^1 F'(x^* + te_k)e_k\,dt = F'(x_k)^{-1}\int_0^1 \left(F'(x_k) - F'(x^* + te_k)\right)e_k\,dt.$$

Korzystając z lipschitzowskości pochodnej i raz jeszcze z lematu 9.2 o oszacowaniu pochodnej, dostajemy

$$\begin{aligned}\|e_{k+1}\| &\leqslant \|F'(x_k)^{-1}\|\int_0^1 \|F'(x_k) - F'(x^* + te_k)\|\,\|e_k\|\,dt \\ &\leqslant 2\|F'(x^*)^{-1}\|\,L\|e_k\|^2\int_0^1 (1-t)\,dt =: C\|e_k\|^2.\end{aligned}$$

Stąd wynika, że jeśli $C\delta < 1$ oraz $x_k \in K(x^*, \delta)$, to także $x_{k+1} \in K(x^*, \delta)$, a więc ciąg (x_k) jest dobrze określony. Co więcej, gdy $C\delta < 1$ to

$$\|e_{k+1}\| \leqslant C\|e_k\|^2 = C\|e_k\|\,\|e_k\| \leqslant C\delta\|e_k\|,$$

a więc ciąg e_k jest zbieżny (co najmniej liniowo — a w rzeczywistości co najmniej kwadratowo) do zera. □

Ćwiczenie 9.2. Wykaż, że jeśli w założeniach standardowych zastąpić warunek lipschitzowskości pochodnej warunkiem

$$\exists \alpha \in (0,1]\ \exists L > 0 \quad \|F'(x) - F'(y)\| \leqslant L\|x-y\|^{\alpha} \qquad \forall x, y \in D,$$

(czyli założyć, że F' jest w swej dziedzinie hölderowska z wykładnikiem α), to metoda Newtona będzie lokalnie zbieżna z wykładnikiem zbieżności co najmniej $1+\alpha$.

Wskazówka. Wystarczy powielić dowód twierdzenia przy standardowych założeniach, modyfikując oszacowanie $\|F'(x_k) - F'(x^* + te_k)\|$.

Ćwiczenie 9.3. Jak, przy założeniach standardowych, oszacować normę błędu, $\|x_k - x^*\|$, przez normę residuum, $\|F(x_k)\|$?

9.3.2. Implementacja metody Newtona

Implementując metodę Newtona, nie będziemy rzecz jasna nigdy *explicite* wyznaczali macierzy odwrotnej $F'(x_k)^{-1}$, tylko rozwiązywali układ równań z macierzą $F'(x_k)$ — tę jednak będziemy już musieli wyznaczyć. Pamiętając także o tym, że metoda Newtona jest metodą iteracyjną, stosując ją będziemy musieli zadbać o postawienie sensownego kryterium zatrzymania metody. Odkładając na bok pytanie o konkretny warunek stopu (por. rozdział 7.5), możemy schematycznie zapisać algorytm realizujący metodę Newtona w następujący sposób:

Schemat metody Newtona

```
function Newton(x, F, stop)
while not stop do begin
  oblicz macierz pochodnej F'(x);
  rozwiąż F'(x)s = F(x);
  x = x − s;
end
return(x);
```

Ćwiczenie 9.4. Macierzowe zadanie własne dla symetrycznej macierzy $A \in \mathbb{R}^{N\times N}$, znajdowania pary $(x,\lambda) \in \mathbb{R}^N \times \mathbb{R}$ spełniającej

$$Ax = \lambda x, \qquad x \neq 0$$

można potraktować jako kwadratowe równanie nieliniowe dla $F : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ danej na przykład wzorem

$$F(x,\lambda) = \begin{pmatrix} Ax - \lambda x \\ 1 - \frac{1}{2}x^T x \end{pmatrix}.$$

Zdefiniuj metodę Newtona dla F. Wykaż, że jeśli λ jest jednokrotną wartością własną, to metoda Newtona dla F będzie lokalnie zbieżna kwadratowo.

Rozwiązanie. Mamy

$$F'(x,\lambda) = \begin{pmatrix} A - \lambda I & -x \\ -x^T & 0 \end{pmatrix}.$$

Zatem metodę Newtona można — jeśli tylko $F'(x,\lambda)$ nieosobliwa — zaimplementować, korzystając na przykład ze wzoru Shermana–Morrisona (por. ćwiczenie 5.23).

Zauważmy, że F jest funkcją gładką, jej pochodna jest lipschitzowska ze stałą Lipschitza równą... no właśnie, ile? Mamy

$$F'(x,\lambda) - F'(y,\mu) = \begin{pmatrix} (\mu-\lambda)I & y-x \\ (y-x)^T & 0 \end{pmatrix},$$

zatem

$$\begin{aligned} \|F'(x,\lambda) - F'(y,\mu)\|_1 &= \max\{|\mu-\lambda| + |(y-x)_1|, \dots, \|y-x\|_1\} \\ &\leqslant \|y-x\|_1 + |\mu-\lambda| = \| \begin{pmatrix} x \\ \lambda \end{pmatrix} - \begin{pmatrix} y \\ \mu \end{pmatrix} \|_1. \end{aligned}$$

A więc — w normie indukowanej normą $\|\cdot\|_1$ — F' jest lipschitzowska ze stałą równą 1. Pozostaje jeszcze sprawdzić, czy w rozwiązaniu — parze własnej x^*, λ^* — macierz pochodnej jest nieosobliwa. Ponieważ dla macierzy symetrycznej $A = Q\Lambda Q^T$, gdzie Λ jest macierzą diagonalną z wartościami własnymi macierzy A, a kolumny macierzy ortogonalnej Q są wektorami własnymi odpowiadającymi tym wartościom własnym, to

$$\begin{pmatrix} Q^T & \\ & 1 \end{pmatrix} F'(x,\lambda) \begin{pmatrix} Q & \\ & 1 \end{pmatrix} = \begin{pmatrix} \Lambda - \lambda I & Q^T x \\ x^T Q & 0 \end{pmatrix}.$$

Niech $\lambda^* = \lambda_i$. Wtedy $x^* = Cq_i$ dla pewnej stałej C. Mamy więc, że $\Lambda - \lambda^* I$ jest macierzą diagonalną, która na diagonali ma zero jedynie na i-tej pozycji (na mocy założenia, że λ^* jest pojedynczą wartością własną) oraz $Q^T x = Ce_i$, gdzie e_i oznacza i-ty wektor jednostkowy. Nietrudno sprawdzić wprost, że taka macierz jest pełnego rzędu.

Ćwiczenie 9.5 (Metoda Schultza wyznaczania macierzy odwrotnej, [15])**.** Niech A będzie macierzą nieosobliwą i rozważmy iterację

$$X_{k+1} = X_k + X_k(I - AX_k),$$

startującą z macierzy kwadratowej X_0. Wykaż, że jeśli $\|I - AX_0\| < 1$, to $X_k \to A^{-1}$ gdy $k \to \infty$.

Wskazówka. Wykaż, że $\|I - AX_{k+1}\| = \|I - AX_k\|^2$, a metoda jest w rzeczywistości metodą Newtona zastosowaną do równania macierzowego $F(X) = A - X^{-1} = 0$.

Przykład 9.4 (Metoda Newtona dla równania Allena–Cahna)**.** Przypomnijmy (zob. przykład 9.1), że równanie nieliniowe, które nas interesuje w przypadku stacjonarnym ma postać

$$F(U_N) = P_N U_N + f(U_N),$$

a w przypadku ewolucyjnym —

$$F(U_N) = U_N + \tau(P_N U_N + f(U_N)).$$

Oba przypadki możemy zatem ogarnąć, definiując funkcję

$$F_{\tau,\alpha}(U_N) = \alpha U_N + \tau(P_N U_N + f(U_N)),$$

która dla $\alpha = 1$ i $\tau \ll 1$ daje nam przypadek ewolucyjny, a dla $\alpha = 0$ i $\tau = 1$ redukuje się do przypadku stacjonarnego równania Allena–Cahna.

Wystarczy więc tylko wyznaczyć pochodną $F'_{\tau,\alpha}(U_N)$. Ponieważ (traktując f jako funkcję na $\mathbb{R}^{N^2}$)

$$f(U_N)_j = \delta u_j(1 - u_j^2) + g_j,$$

to

$$\frac{\partial f_j}{\partial u_k}(U_N) = \begin{cases} 1 - 3u_j^2, & k = j, \\ 0, & k \neq j, \end{cases}$$

zatem ostatecznie

$$F'_{\tau,\alpha}(U_N)\, V = (\alpha I + \tau(P_N + \text{diag}(1 - 3u_j^2)))\, V.$$

Przykład 9.5 (Numeryczne eksperymenty z metodą Newtona dla równania Allena–Cahna)**.** Dla uproszczenia, w eksperymentach numerycznych dotyczących działania różnych metod rozwiązywania równania Allena–Cahna opisywanego w przykładzie 9.1, będziemy rozważać dyskretyzacje *jednowymiarowego* stacjonarnego równania Allena–Cahna,

$$(T_N u)_j + \delta u_j(1 - u_j^2) = g_j, \qquad j = 1, \ldots, N,$$

gdzie $\delta = 10$ oraz $g_j = j/(N+1)$.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, resid, info, output] = newton(nazwa_f, nazwa_df, x0, atol, rtol, step, maxit)
%
% [x, resid, info, output] = newton(nazwa_f, x0, atol, rtol, step, maxit)
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch9.

Ćwiczenie 9.6. Przeprowadź podobne eksperymenty numeryczne dla równania Allena–Cahna w kwadracie, jak to opisywano we wcześniejszych przykładach.

10. Wariacje na temat metody Newtona

Podstawowy element metody Newtona — rozwiązywanie równania zlinearyzowanego — w przypadku, gdy N jest duże, może stanowić wąskie gardło całego procesu iteracyjnego. Dlatego w tym i następnym rozdziale poszukamy skutecznych metod ominięcia tego ograniczenia; jednak na początek przytoczymy inną wersję twierdzenia o zbieżności, która (przy silniejszych założeniach) zagwarantuje nam także *istnienie* rozwiązań.

10.1. Twierdzenie o istnieniu rozwiązań

W dotychczasowej analizie zakładaliśmy, że rozwiązanie x^* równania $F(x) = 0$ istnieje. Okazuje się, że nieco modyfikując założenia standardowe można udowodnić i istnienie rozwiązania, i lokalną zbieżność metody Newtona do tego rozwiązania.

Twierdzenie 10.1 (Kantorowicza, o istnieniu i zbieżności)**.** *Niech $F : \mathbb{R}^N \supset D \to \mathbb{R}^N$, przy czym D jest niepusty, otwarty i wypukły, i niech F będzie różniczkowalna w D oraz istnieje $L > 0$ takie, że*

$$\|F'(x) - F'(y)\| \leqslant L\|x - y\| \qquad \forall x, y \in D.$$

Niech istnieje $x_0 \in D$ takie, że $F'(x_0)$ jest nieosobliwa oraz dla pewnych dodatnich β, η zachodzi

$$\|F'(x_0)^{-1}F(x_0)\| \leqslant \beta, \qquad \|F'(x_0)^{-1}\| \leqslant \eta,$$

przy czym $\alpha = L\beta\eta \leqslant 1/2$. Określmy $\delta = (1-\sqrt{1-2\alpha})/(L\beta)$ i załóżmy, że $\bar{K}(x_0, \delta) \subset D$.

Wówczas ciąg x_k zdefiniowany metodą Newtona jest dobrze określony, co więcej $x_k \in \bar{K}(x_0, \delta)$ oraz ma granicę x^*, która jest jedynym miejscem zerowym F w $\bar{K}(x_0, \delta)$. Ponadto

$$\|x_k - x^*\| \leqslant (L\beta 2^k)(2\alpha)^{2^k}$$

dla $k \geqslant 0$.

Dowód. Dowód tego twierdzenia można znaleźć np. w [12, rozdział 12.6.1]. □

10.2. Obniżanie kosztu iteracji

Z punktu widzenia praktycznej realizacji, każdy krok metody Newtona ma parę potencjalnie kosztownych momentów, które teraz przedyskutujemy.

— Wyznaczenie wzoru na $F'(x)$ może być trudne: na przykład, gdy F jest zadana bardzo skomplikowaną procedurą (a nie jednolinijkowym wzorem), wtedy ani ręczne, ani automatyczne

różniczkowanie F może nie prowadzić do poprawnych czy zadowalających rezultatów. Może też być bardzo żmudne.

— Wyznaczenie wartości $F(x)$, a zwłaszcza: $F'(x)$, może być kosztowne. Rzeczywiście, $F'(x)$ to N^2 elementów macierzy pochodnej, a dodatkowo — poza najprostszymi przypadkami — pochodna jest zwykle dana bardziej skomplikowanym wyrażeniem niż sama funkcja. Rozsądne jest więc przyjęcie[1], że w praktyce koszt wyznaczenia $F'(x)$ będzie rzędu $c \cdot N \cdot$ (koszt obliczenia $F(x)$), z raczej dużą stałą c. Nawet wyznaczenie pojedynczej wartości $F(x)$ może być w niektórych zadaniach bardzo kosztowne. W [3] wspomina się o $F : \mathbb{R}^{50} \to \mathbb{R}^{50}$, której jedna wartość kosztowała 100 godzin pracy ówczesnego szybkiego komputera.

— Rozwiązanie układu z $F'(x)$ jest potencjalnie najbardziej kosztownym fragmentem iteracji. Rzeczywiście, gdy $F'(x)$ jest macierzą gęstą bez żadnych szczególnych własności, to koszt rozwiązania układu

$$F'(x)s = F(x)$$

jest rzędu $O(N^3)$.

Te obserwacje prowadzą do kilku heurystycznych sposobów modyfikacji metody Newtona tak, by obniżyć koszt jednej jej iteracji. Czasem — choć nie zawsze, i dlatego te sposoby są ważne! — obniżenie kosztu iteracji będzie skutkowało pogorszeniem szybkości zbieżności metody, ale nawet i wówczas może okazać się, że ostatecznie metoda wolniej zbieżna, ale tańsza, będzie bardziej efektywna od metody Newtona.

10.2.1. Uproszczona metoda Newtona

W uproszczonej metodzie Newtona, najdroższy fragment iteracji Newtona — wyznaczenie rozkładu trójkątnego macierzy pochodnej — usuwamy poza pętlę:

$$x_{k+1} = x_k - F'(x_0)^{-1}F(x_k).$$

Rzeczywiście, jeśli bowiem na samym początku iteracji dokonamy kosztem $O(N^3)$ flopów rozkładu LU (lub QR) macierzy $F'(x_0)$ i zapamiętamy czynniki rozkładu, to potem każdy z układów równań:

$$F'(x_0)s = F(x_k)$$

będziemy mogli rozwiązać już tylko kosztem nie wyższym niż $O(N^2)$! Oznacza to, że koszt jednej iteracji spada nam do $O(N^2)$, jest więc N-krotnie mniejszy niż w przypadku oryginalnej metody Newtona. Ceną za przyspieszenie jest wyraźne zmniejszenie szybkości zbieżności metody:

Twierdzenie 10.2 (o zbieżności uproszczonej metody Newtona)**.** *Przy standardowych założeniach, istnieją stałe dodatnie C i δ takie, że jeśli $x_0 \in K(x^*, \delta)$, to uproszczona metoda Newtona jest zbieżna przynajmniej liniowo do x^* oraz*

$$\|x_{k+1} - x^*\| \leqslant C\|x_0 - x^*\|\|x_k - x^*\| \qquad \forall k \in \mathbb{N}. \tag{10.1}$$

Dowód. Zostanie podany później, gdy będziemy dysponowali wygodnym narzędziem do badania tego typu metod. □

[1] O ile $F'(x)$ nie jest macierzą rozrzedzoną.

Ćwiczenie 10.1. Zaimplementuj uproszczoną metodę Newtona.

Rozwiązanie. Wystarczy zmodyfikować algorytm realizujący metodę Newtona:

Uproszczona metoda Newtona

```
function simplerNewton(x, F, stop)
oblicz macierz pochodnej F'(x);
wyznacz rozkład F'(x), np. QR = F'(x);
while not stop do begin
  rozwiąż układ z macierzą trójkątną, Rs = Q^T F(x);
  x = x − s;
end
return(x);
```

Ćwiczenie 10.2. Metoda Szamańskiego to metoda, w której po m krokach uproszczonej metody Newtona dokonujemy wyznaczenia i rozkładu macierzy pochodnej (czyli restartujemy metodę uproszczoną, biorąc za przybliżenie początkowe ostatnio wyznaczone przybliżenie rozwiązania). Jest to więc coś pośredniego pomiędzy prawdziwą metodą Newtona (w której w każdej iteracji wyznacza się i rozkłada macierz pochodnej) a uproszczoną metodą Newtona (gdzie macierz pochodnej wyznacza się tylko jeden raz).

Formalnie możemy więc przyjąć, że m kroków uproszczonej metody Newtona to jest „jeden duży krok" i określić iterację Szamańskiego następująco:

$$\begin{aligned} x_{k+\frac{1}{m}} &= x_k - F'(x_k)^{-1}F(x_k) \\ x_{k+\frac{j+1}{m}} &= x_{k+\frac{j}{m}} - F'(x_k)^{-1}F(x_{k+\frac{j}{m}}), \qquad j = 1, \ldots, m-1. \end{aligned}$$

Znajdź wykładnik p taki, że

$$\|x_{k+1} - x^*\| \leqslant C\|x_k - x^*\|^p.$$

Rozwiązanie. Z definicji metody, x_{k+1} wyznaczamy jako m-tą iterację uproszczonej metody Newtona startującej z x_k. Na mocy oszacowania błędu uproszczonej metody Newtona mamy więc

$$\|x_{k+\frac{j+1}{m}} - x^*\| \leqslant C\|x_k - x^*\|\|x_{k+\frac{j}{m}} - x^*\|,$$

skąd

$$\|x_{k+1}-x^*\| \leqslant C\|x_k-x^*\|\|x_{k+\frac{m-1}{m}}-x^*\| \leqslant (C\|x_k-x^*\|)^2\|x_{k+\frac{m-2}{m}}-x^*\| \leqslant \cdots \leqslant C^m\|x_k-x^*\|^{m+1}.$$

A więc metoda ma wykładnik $m+1$.

Ćwiczenie 10.3. Porównaj efektywność metody Szamańskiego i Newtona w zależności od rozmiaru zadania N, przy następujących modelowych założeniach:

— koszt wyznaczenia $F(x)$ jest równy c_0N
— koszt wyznaczenia $F'(x)$ jest równy c_1N^2
— koszt wyznaczenia rozkładu $F'(x)$ jest równy c_rN^3
— koszt wyznaczenia rozwiązania układu z macierzą $F'(x)$ o danym rozkładzie jest równy c_sN^2

Rozwiązanie. Oczywiście musimy założyć $m > 1$, w przeciwnym razie obie metody są tożsame. Dla metody Newtona mamy

$$E_{\text{Newton}} = 2^{1/(c_r N^3 + (c_1+c_s)N^2 + c_0 N)},$$

a dla metody Szamańskiego

$$E_{\text{Szamański}} = (m+1)^{1/(c_r N^3 + (c_1+mc_s)N^2 + mc_0 N)}.$$

Stąd $E_{\text{Szamański}} \geqslant E_{\text{Newton}}$ wtedy i tylko wtedy, gdy $\log_2 E_{\text{Szamański}} \geqslant \log_2 E_{\text{Newton}}$. Dalej przekształcając, dostajemy warunek wiążący ze sobą m i N:

$$\log_2(m+1) \geqslant \frac{1 + m\,O(\frac{1}{N})}{1 + O(\frac{1}{N})},$$

co zachodzi dla N dostatecznie dużych względem m.

Ćwiczenie 10.4 (liniowa niezmienniczość iteracji Newtona)**.** Rozważmy równanie $F(x) = 0$ i jego liniową transformację

$$G(y) = AF(B^{-1}y) = 0,$$

gdzie A, B są nieosobliwymi macierzami rozmiaru $N \times N$. Wykaż, że jeśli tylko $y_0 = Bx_0$, to iteracja Newtona dla G,

$$y_{k+1} = y_k - G'(y_k)^{-1} G(y_k)$$

jest równoważna metodzie Newtona dla F w tym sensie, że zawsze zachodzi $y_k = Bx_k$.

Rozwiązanie. Iteracja Newtona dla F to

$$x_{k+1} = x_k - F'(x_k)^{-1} F(x_k).$$

Mnożąc lewostronnie przez B i podstawiając gdzie trzeba $x_k = B^{-1}y_k$ mamy

$$y_{k+1} = y_k - BF'(B^{-1}y_k)^{-1} F(B^{-1}y_k) = y_k - BF'(B^{-1}y_k)^{-1} A^{-1}\, AF(B^{-1}y_k).$$

Na zakończenie wystarczy zauważyć, że $BF'(B^{-1}y_k)^{-1}A^{-1} = (AF'(B^{-1}y_k)B^{-1})^{-1} = G'(y_k)^{-1}$.

Ćwiczenie 10.5. Wykaż, że jeśli spełnione są założenia standardowe oraz metoda Newtona jest zbieżna, to dla dostatecznie dużych k zachodzi

$$\|F'(x_k)^{-1} F(x_{k+1})\| \leqslant \frac{1}{4} \|F'(x_k)^{-1} F(x_k)\|.$$

Może to więc być dość prosty test (niezmienniczy ze względu na liniowe transformacje zmiennej niezależnej i zależnej!), pozwalający *przypuszczać*, że *nie zachodzi* zbieżność: na przykład, gdy na pewnym kroku metody stwierdzimy $\|F'(x_k)^{-1} F(x_{k+1})\| \geqslant \frac{3}{4} \|F'(x_k)^{-1} F(x_k)\|$, możemy wtedy uznać *prawdopodobny brak zbieżności* i zakończyć iterację.

10.2.2. Wpływ niedokładności wyznaczenia pochodnej lub funkcji na zbieżność metody Newtona

Twierdzenie 10.3 (o lokalnej stabilności metody Newtona)**.** *Przy standardowych założeniach, istnieją stałe dodatnie C, δ oraz $\tilde{\delta}$ takie, że jeśli $x_k \in K(x^*, \delta)$ oraz $\|\Delta_k\| < \tilde{\delta}$, to*

$$x_{k+1} = x_k - \left(F'(x_k) + \Delta_k\right)^{-1} \left(F(x_k) + \epsilon_k\right) \tag{10.2}$$

jest dobrze określony oraz

$$\|e_{k+1}\| \leqslant C \left(\|e_k\|^2 + \|\Delta_k\|\|e_k\| + \|\epsilon_k\|\right), \tag{10.3}$$

gdzie $e_k = x_k - x^*$.

Dowód. Dowód znajdziemy np. w [9]. □

Z twierdzenia o lokalnej stabilności metody Newtona można wywnioskować wiele twierdzeń o zbieżności tych modyfikacji metody Newtona, które sprowadzają się do zaburzenia oryginalnej metody Newtona, na przykład twierdzenia o zbieżności uproszczonej metody Newtona.

Dowód. (Twierdzenia 10.2, o zbieżności uproszczonej metody Newtona.) Jeden krok uproszczonej metody Newtona

$$x_{k+1} = x_k - F'(x_0)^{-1} F(x_k)$$

możemy zapisać w języku twierdzenia o lokalnej stabilności metody Newtona, gdzie

$$\Delta_k = F'(x_0) - F'(x_k), \qquad \epsilon_k = 0.$$

Niech więc δ będzie dostatecznie małe tak, by dla x_k zachodziło twierdzenie o lokalnej stabilności metody Newtona w kuli $K(x^*, \delta)$. Dodatkowo załóżmy, że $x_0 \in K(x^*, \delta)$. Wtedy

$$\|\Delta_k\| = \|F'(x_0) - F'(x_k)\| \leqslant L\|x_0 - x_k\| \leqslant L\left(\|x_0 - x^*\| + \|x_k - x^*\|\right)$$

i na mocy właśnie tw. o lokalnej stabilności,

$$\|e_{k+1}\| \leqslant \tilde{C}\left(\|e_k\|^2 + L\left(\|e_0\| + \|e_k\|\right)\|e_k\|\right) \leqslant C\|e_0\|\|e_k\|.$$

Ponieważ z założenia $\|e_0\| < \delta$, to z powyższego wynika, że

$$\|e_{k+1}\| \leqslant C\delta\|e_k\|$$

jeśli więc δ jest na tyle małe, by dodatkowo $C\delta < 1$, to ciąg x_k zawiera się w $K(x^*, \delta)$ oraz e_k musi być zbieżny do zera przynajmniej liniowo. □

10.2.3. Metoda z przybliżoną pochodną

Jak już wspominaliśmy, innym kłopotliwym momentem w realizacji metody Newtona jest wyznaczanie macierzy pochodnej. Opracowanie dokładnej formuły obliczania $F'(x)$ może być żmudne, podatne na ludzkie pomyłki. Uzyskany wzór może być ciężki w implementacji i kosztowny w użyciu.

Ale, ponieważ metoda Newtona jest jedynie metodą przybliżoną, można zaryzykować użycie *przybliżonej pochodnej* — najlepiej: przybliżonej niskim kosztem. Jednym z pomysłów mogłoby być wyznaczenie przybliżenia pochodnej na podstawie ilorazów różnicowych, zgodnie z poniższym twierdzeniem.

Twierdzenie 10.4 (o różnicowej aproksymacji pochodnej kierunkowej)**.** *Przyjmijmy, że są spełnione założenia standardowe. Niech x będzie ustalonym punktem w D i niech $w \in \mathbb{R}^N$. Niech $h \in \mathbb{R}$ będzie na tyle małe, by $x + hw \in D$. Wtedy*

$$\| \frac{F(x+hw) - F(x)}{h} - F'(x)w \| \leqslant \frac{L}{2} \|w\|^2 \cdot |h|.$$

Twierdzenie to gwarantuje więc, że pochodną w kierunku wektora w można aproksymować ilorazem różnicowym

$$F'(x)w \approx \frac{F(x+hw) - F(x)}{h}$$

z błędem rzędu $O(h)$.

Dowód. Twierdzenie jest prostą konsekwencją twierdzenia o wartości średniej i założenia lipschitzowskości pochodnej:

$$\begin{aligned} \frac{1}{h}\left(F(x+hw) - F(x)\right) &= \frac{1}{h}\int_0^1 F'(x+thw)\cdot hw\,dt = \int_0^1 F'(x)w + (F'(x+thw) - F'(x))w\,dt \\ &= F'(x)w + \int_0^1 (F'(x+thw) - F'(x))w\,dt. \end{aligned}$$

Stąd

$$\| \frac{1}{h}\left(F(x+hw) - F(x)\right) - F'(x)w \| \leqslant \int_0^1 \|F'(x+thw) - F'(x)\|\|w\|\,dt \leqslant \int_0^1 \|thw\|\|w\|\,dt = \frac{L}{2}\|w\|^2 \cdot |h|.$$

□

Z powyższego wynika pomysł na przybliżenie całej macierzy pochodnej ilorazami różnicowymi: wystarczy wziąć $F'(x_k) \approx A_k$, gdzie j-ta kolumna A_k, $(A_k)_j$, jest wyznaczona jako iloraz różnicowy

$$(A)_j = \frac{1}{h}\left(F(x+he_j) - F(x)\right) \approx F'(x)e_j.$$

Twierdzenie 10.5 (o zbieżności metody z pochodną przybliżoną ilorazami różnicowymi)**.** *Przy standardowych założeniach, istnieją dodatnie δ i h (dostatecznie małe) takie, że jeśli (h_k) jest ciągiem takim, że $0 < |h_k| \leqslant h$ oraz $x_0 \in K(x^*, \delta)$, to ciąg zdefiniowany wzorem*

$$x_{k+1} = x_k - A_k^{-1} F(x_k),$$

w którym kolumny macierzy A_k zadane są ilorazami różnicowymi:

$$(A_k)_j = \frac{1}{h}\left(F(x_k + h_k e_j) - F(x_k)\right), \qquad j = 1, \ldots, N,$$

jest dobrze określony i zbieżny przynajmniej liniowo do x^*. Co więcej,

- *jeśli $h_k \to 0$, to $x_k \to x^*$ superliniowo;*
- *jeśli $|h_k| \leqslant M\|x_k - x^*\|$ dla pewnej stałej M, to $x_k \to x^*$ przynajmniej kwadratowo;*
- *jeśli $|h_k| \leqslant M\|F(x_k\|$ dla pewnej stałej M, to $x_k \to x^*$ przynajmniej kwadratowo.*

Dowód. Dla wygody, dowód poprowadzimy w normie $\|\cdot\|_1$ — tzw. kolumnowej normie macierzowej (w przestrzeni skończenie wymiarowej i tak wszystkie normy są równoważne). Mamy więc

$$\|A_k\|_1 = \max_j \|(A_k)_j\|_1,$$

oraz

$$\|A_k - F'(x_k)\|_1 = \max_j \|(A_k)_j - F'(x_k)e_j\|_1.$$

Tymczasem, na mocy lematu o różnicowej aproksymacji pochodnej,

$$\|(A_k)_j - F'(x_k)e_j\|_1 \leqslant \frac{L}{2}\|e_j\|_1^2 |h_k| = \frac{L}{2}|h_k|.$$

Oznaczmy $\Delta_k = A_k - F'(x_k)$, wtedy $\|\Delta_k\|_1 \leqslant \dfrac{L}{2}|h_k|$. Na mocy twierdzenia o lokalnej stabilności metody Newtona z (dla δ, h dostatecznie małych) mamy, że A_k jest nieosobliwa oraz zachodzi

$$\|e_{k+1}\|_1 \leqslant C\left(\|e_k\|_1^2 + |h_k|\|e_k\|_1\right).$$

Stąd już wynika teza twierdzenia. Ostatni warunek kwadratowej zbieżności wynika z oszacowania residuum przez błąd,

$$\|F(x_k)\|_1 \leqslant 2\|F'(x^*)\|_1 \|e_k\|_1,$$

gdy tylko x_k jest dostatecznie blisko x^*. □

10.2.4. Niedokładna metoda Newtona

Jak pamiętamy, najkosztowniejszą częścią jednej iteracji Newtona jest rozwiązywanie układu równań z macierzą pochodnej. Alternatywą dla metody bezpośredniej rozwiązywania równania poprawki

$$F'(x_k)s = F(x_k)$$

mogłaby być, zwłaszcza w przypadku gdy N jest duże, metoda iteracyjna (mielibyśmy zatem do czynienia z iteracją wewnątrz iteracji). Na k-tym kroku metody Newtona moglibyśmy zatrzymywać wewnętrzną iterację stosując np. residualne kryterium stopu z parametrem wymuszającym η_k,

$$\|F(x_k) - F'(x_k)s\| \leqslant \eta_k \|F(x_k)\|.$$

Taką modyfikację metody Newtona nazywa się niedokładną metodą Newtona.

Twierdzenie 10.6 (o błędzie niedokładnej metody Newtona)**.** *Przy standardowych założeniach, istnieją dodatnie stałe C i δ takie, że jeśli $x_k \in K(x^*, \delta)$ oraz s spełnia warunek*

$$\|F'(x_k)s - F(x_k)\| \leqslant \eta_k \|F(x_k)\|,$$

to następna iteracja niedokładnej metody Newtona dana wzorem

$$x_{k+1} = x_k - s$$

spełnia

$$\|e_{k+1}\| \leqslant C\left(\|e_k\| + \eta_k\right)\|e_k\|.$$

Dowód. Niech δ będzie na tyle małe, by zachodził lemat 9.2 o oszacowaniu funkcji i pochodnej, a także by zachodziło twierdzenie 9.3 o zbieżności metody Newtona z punktem startowym x_k. Aby udowodnić twierdzenie, najpierw postaramy się wyłuskać związek pomiędzy naszą metodą, a prawdziwą metodą Newtona (o której już sporo wiemy). Mamy bowiem

$$x_{k+1} = x_k - s = x_k - F'(x_k)^{-1}F(x_k) + F'(x_k)^{-1}F(x_k) - s = x_{k+1}^{\text{Newton}} + F'(x_k)^{-1}r,$$

gdzie r jest residuum rozwiązania równania na poprawkę,

$$r = F(x_k) - F'(x_k)s.$$

Stąd wynika, że

$$x_{k+1} - x^* = x_{k+1}^{\text{Newton}} - x^* + F'(x_k)^{-1}r$$

i konsekwentnie

$$\|x_{k+1} - x^*\| \leqslant \|x_{k+1}^{\text{Newton}} - x^*\| + \|F'(x_k)^{-1}r\|.$$

Ponieważ na mocy twierdzenia o zbieżności metody Newtona mamy $\|x_{k+1}^{\text{Newton}} - x^*\| \leqslant \tilde{C}\|x_k - x^*\|^2$, to wystarczy oszacować ostatni człon nierówności. Z definicji r mamy

$$\|F'(x_k)^{-1}r\| \leqslant \|F'(x_k)^{-1}\|\|F(x_k) - F'(x_k)s\| \leqslant 2\|F'(x^*)^{-1}\| \cdot \eta_k \|F(x_k)\| \leqslant 4\,\mathrm{cond}(F'(x^*))\eta_k\|x_k - x^*\|.$$

(W ostatniej nierówności skorzystaliśmy z lematu o oszacowaniu funkcji i pochodnej.) □

W powyższym dowodzie dostaliśmy rezultat, w którym stała C w oszacowaniu błędu zależała od uwarunkowania $F'(x^*)$, skąd moglibyśmy wysnuć wniosek, że jeśli $F'(x^*)$ jest źle uwarunkowana, to η_k należy brać patologicznie małe. W rzeczywistości nie jest aż tak źle i można osłabić założenia na ciąg (η_k), ale pod warunkiem zmiany normy, w której badamy błąd, tak, by uwzględniać zachowanie się pochodnej: $\|\cdot\| = \|F'(x^*)\cdot\|$ (zob. [9]).

Wniosek 10.1 (twierdzenie o zbieżności niedokładnej metody Newtona)**.** *Przy standardowych założeniach, istnieją δ i η (dostatecznie małe) takie, że jeśli $x_0 \in K(x^*, \delta)$ oraz $0 \leqslant \eta_k \leqslant \eta$ to niedokładna metoda Newtona z parametrem wymuszającym η_k jest zbieżna przynajmniej liniowo do x^*. Ponadto,*

- *jeśli $\eta_k \to 0$, to zbieżność jest superliniowa;*
- *jeśli $\eta_k \leqslant M_\eta\|F(x_k)\|$ dla pewnego ustalonego M_η, to zbieżność jest przynajmniej kwadratowa.*

Ćwiczenie 10.6. Udowodnij powyższy wniosek.

Niedokładna metoda Newtona jest bardzo popularną metodą w przypadku, gdy N jest bardzo duże: wówczas trudno myśleć o sposobach rozwiązywania równania poprawki innych niż jakaś metoda iteracyjna. Dodatkowym plusem tej metody jest to, że znakomicie można ją łączyć z przybliżaniem pochodnej kierunkowej ilorazami różnicowymi; rzeczywiście, głównym składnikiem metody iteracyjnej jest mnożenie $F'(x_k)w$, co można tanio przybliżać w sposób opisany w rozdziale 10.2.3. W ten sposób oszczędzamy na kilku polach na raz:
— nie musimy wykonywać, zazwyczaj bardzo kosztownego, rozkładu macierzy pochodnej;
— nie musimy nawet wyznaczać macierzy pochodnej, lecz w zamian wystarczy, że w każdej wewnętrznej iteracji, jedną dodatkową wartość $F(x_k + hw)$;
— ponieważ po kilku iteracjach zbliżamy się do rozwiązania, że $x_{k+1} \approx x_k$, w ten sposób od razu dysponujemy dobrym przybliżeniem startowym dla iteracji wewnętrznej: w sprzyjających okolicznościach metody Kryłowa mogą z tego zrobić dodatkowy pożytek;
— możemy ograniczyć koszt rozwiązywania układu z macierzą pochodnej (liczbę iteracji wewnętrznych), stosując łagodne kryterium stopu, gdy jesteśmy daleko od rozwiązania.

Przykład 10.1 (Numeryczne eksperymenty z wariantami metody Newtona dla równania Allena–Cahna)**.** Porównamy jakość pracy kilku metod:

— klasycznej metody Newtona
— uproszczonej metody Newtona
— metody Szamańskiego o $m = 4$ krokach
— metody Newtona z pochodną przybliżoną ilorazami różnicowymi (ze stałym, ale bardzo małym krokiem, $h \approx \sqrt{\epsilon}$, gdzie ϵ to precyzja arytmetyki.

Do porównania wybierzemy:
— wykresy uzyskanego rozwiązania, $u(x)$, dla każdej z metod
— wykresy residuum, $|F(u(x))|$, dla każdej z metod
— historię zbieżności metody, $\|F(u_j)\|_2$
— liczbę iteracji, czas pracy, końcową wartość residuum.

```
%
% rozwiazania zadania z macierza rozrzedzona
%
disp('Matematyka obliczeniowa II');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, resid, info, output] = newton(nazwa_f, nazwa_df, x0, atol, rtol, step, maxit)
%
% [x, resid, info, output] = newton(nazwa_f, x0, atol, rtol, step, maxit)
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch10.

Zwróć uwagę na to, że odpowiednio dobierając częstość uaktualniania macierzy pochodnej, można znacząco poprawić efektywność metody (w porównaniu do metody Newtona). Z drugiej strony, zbyt rzadka aktualizacja może przeszkodzić w uzyskaniu zbieżności, jak to dzieje się w naszym przykładzie w przypadku uproszczonej metody Newtona.

Ćwiczenie 10.7. Sprawdź, jak zmienią się wyniki (uzyskane rozwiązanie, jego dokładność, a także — koszt metody i szybkość jej zbieżności), gdy zmienisz jeden z poniższych parametrów
— osłabisz nieliniowość, biorąc $\delta = 1$,

— nieco wzmocnisz nieliniowość, biorąc $\delta = 15$,
— zwiększysz rozmiar zadania N do 400,
— zmniejszysz N do 20,
— zmniejszysz tolerancję błędu do 10^{-12}.

Wskazówka. Zdaje się, że dla $N = 20$ mamy niejednoznaczność rozwiązania? Jak to zweryfikować?

Ćwiczenie 10.8. Zaimplementuj niedokładną metodę Newtona i wykorzystaj ją w skrypcie rozwiązującym (najlepiej: *dwuwymiarowe*) równanie Allena–Cahna. Jak wpłynęło to na efektywność metody, w porównaniu do standardowej metody Newtona?

Przykład 10.2. Monografii Kelley'a towarzyszy zestaw skryptów, pozwalających przetestować działanie różnych metod iteracyjnych w kilku praktycznych zadaniach [9].

Poniższy skrypt, którego podstawowe kody źródłowe pochodzą ze strony http://www4.ncsu.edu/~ctk/newton, umożliwi Ci zapoznanie się z zadaniem rozwiązania równania Chandrasekhara [9, rozdział 5.6].

```
disp('********** Using C.T.Kelley functions, see the URL **********');

baseurl = 'http://www4.ncsu.edu/~ctk/newton/SOLVERS/';
files = {'nsold.m','nsoli.m','brsola.m'};
for i = 1:length(files)
  url = [baseurl,files{i}]
  S = urlread(url);
  fid = fopen(files{i}, 'w');
  fprintf(fid, '%s', S);
  fclose(fid);
```

To tylko fragment skryptu Octave. Możesz go uruchomić na http://mst.mimuw.edu.pl/lecture.php?lecture=mo2&part=Ch10.

Aby w pełni wykorzystać możliwości skryptu, zachęcamy do uruchomienia go na własnym (podłączonym do internetu) komputerze i następnie do wnikliwej obserwacji zmiany wyników przy zmianie, czy to parametrów zadania, czy to parametrów pracy solvera.

11. Metoda Broydena

W przypadku jednowymiarowym, metoda siecznych:

$$x_{k+1} = x_k - \left(\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}\right)^{-1} f(x_k)$$

jest interesującą alternatywą dla metody Newtona: jej zbieżność jest superliniowa (dokładniej: wykładnicza z wykładnikiem $p = \frac{1+\sqrt{5}}{2} \approx 1.61\ldots$ — a więc tylko trochę gorsza niż metody stycznych), a za to istotny koszt jednej iteracji ogranicza się do wyznaczenia tylko *jednej* nowej wartości funkcji! (Pochodna jest niepotrzebna).

Uogólnienie pomysłu metody siecznych na przypadek wielowymiarowy tak, by zachować jej dwie główne cechy: superliniową zbieżność oraz niski koszt iteracji (nie wymagający w szczególności wyznaczania pochodnej ani obliczania dodatkowych wartości funkcji) nie jest trywialne i zostało zrealizowane w metodzie Broydena. Iterację Broydena określimy wzorem

$$x_{k+1} = x_k - A_k^{-1} F(x_k),$$

gdzie $A_0 = F'(x_0)$ (lub jest jej przybliżeniem wyznaczonym za pomocą ilorazów różnicowych), a kolejne macierze A_k wyznaczamy według wzoru

$$A_{k+1} = A_k + \frac{1}{s_k^T s_k} F(x_{k+1}) s_k^T,$$
$$s_k = x_{k+1} - x_k.$$

(Zauważmy, że s_k jest jawnie wyznaczane w trakcie algorytmu, gdyż s_k jest rozwiązaniem równania poprawki, $A_k s_k = -F(x_k)$.)

Jak widzimy z powyższej definicji, koszt jednej iteracji metody Broydena jest niewygórowany. Rzeczywiście, jeśli z poprzedniego kroku znamy rozkład LU lub QR macierzy A_{k-1}, to wyznaczenie rozkładu A_k, która jest postaci $A_{k+1} = A_k + u_k v_k^T$, jest zadaniem *wyznaczenia rozkładu dla modyfikacji macierzy rzędu 1* — a na to są znane (tanie!) algorytmy [7], [10], działające kosztem $O(N^2)$ flopów[1], o czym szerzej mówimy w poniższym rozdziale.

11.1. Realizacja metody Broydena

Oczywiście, gdybyśmy tylko skorzystali wprost ze wzoru

$$A_{k+1} = A_k + \frac{1}{s_k^T s_k} F(x_{k+1}) s_k^T,$$

to co prawda wyznaczenie A_k kosztowałoby nas tylko $O(N^2)$ flopów, ale za to rozkład macierzy A_k musiałby nas kosztować $O(N^3)$. Jednak, jako że macierz A_{k+1} różni się od A_k tylko o

[1] W metodzie Broydena nie ma więc — wbrew pozorom — potrzeby jawnego wyznaczania macierzy A_k!

macierz rzędu jeden, będziemy mogli od razu przykładać *odwrotność* macierzy A_k do wektora na podstawie rozkładu macierzy A_0, kosztem jedynie $O(N^2 + kN)$.

Algorytm działający w oparciu o aktualizację macierzy odwrotnej do A_k może być mniej stabilny numerycznie od algorytmu używającego aktualizacji rozkładu QR (zob. [10, 10.3.1, Algorytm 2]) macierzy A_k i dlatego wskazuje się sposoby kontrolowania przebiegu pracy tej metody [4], którymi tutaj nie będziemy się zajmować.

Aby tanio odwracać kolejne macierze A_k, wykorzystamy wzór Shermana–Morrisona, którego wyprowadzeniu było poświęcone ćwiczenie 5.24.

Stwierdzenie 11.1 (wzór Shermana–Morrisona)**.** *Niech $B = A + uv^T$, gdzie A jest macierzą nieosobliwą $N \times N$ oraz $u, v \in \mathbb{R}^N$. B jest nieosobliwa wtedy i tylko wtedy, gdy $1 + v^T A^{-1} u \neq 0$ i wówczas*

$$B^{-1} = \left(I - \frac{(A^{-1}u)v^T}{1 + v^T A^{-1} u} \right) A^{-1}.$$

U nas

$$A_{k+1} = A_k + u_k v_k^T,$$

gdzie $u_k = F(x_{k+1})/\|s_k\|_2$, $v_k = s_k/\|s_k\|_2$. Oznaczając dodatkowo

$$z_k = A_k^{-1} u_k = A_k^{-1} F(x_{k+1})/\|s_k\|_2, \qquad w_k = \frac{z_k}{1 + v_k^T z_k},$$

ze wzoru Shermana–Morrisona mamy więc

$$A_{k+1}^{-1} = (I - w_k v_k^T) A_k^{-1} = \ldots = (I - w_k v_k^T) \cdots (I - w_0 v_0^T) A_0^{-1}, \tag{11.1}$$

Wydawałoby się więc, że by skorzystać z powyższego wzoru, należałoby pamiętać dodatkowe wektory u_j, w_j, v_j dla $j = 0, \ldots, k$. W rzeczywistości jednak możemy dalej wymasować powyższe zależności i wyeliminować te wektory, ograniczając się jedynie do pamiętania sekwencji poprawek $s_0, \ldots, s_k$.

Rzeczywiście, w metodzie Broydena interesuje nas przecież nie sama macierz A_{k+1}^{-1}, tylko nowa wartość poprawki, $s_{k+1} = -A_{k+1}^{-1} F(x_{k+1})$. Tymczasem, korzystając z definicji z_k i v_k,

$$-s_{k+1} = A_{k+1}^{-1} F(x_{k+1}) = (I - w_k v_k^T) \underbrace{A_k^{-1} F(x_{k+1})}_{= \|s_k\|_2 z_k} = \|s_k\|_2 (I - \frac{z_k}{1 + v_k^T z_k} v_k^T) z_k$$

$$= \|s_k\|_2 z_k (1 - \frac{v_k^T z_k}{1 + v_k^T z_k}) = \|s_k\|_2 z_k \frac{1}{1 + v_k^T z_k} = \|s_k\|_2 w_k.$$

Stąd dostajemy elegancką zależność $w_k = -s_{k+1}/\|s_k\|_2$, którą możemy podstawić z powrotem do (11.1) i otrzymać

$$-s_{k+1} = A_{k+1}^{-1} F(x_{k+1}) = \left(I + \frac{s_{k+1} s_k^T}{\|s_k\|_2^2} \right) A_k^{-1} F(x_{k+1}). \tag{11.2}$$

Na tej podstawie (s_{k+1} występuje po obu stronach tej równości!) konkludujemy, że

$$-s_{k+1} = A_k^{-1} F(x_{k+1}) / \left(1 + \frac{s_k^T A_k^{-1} F(x_{k+1})}{\|s_k\|_2^2} \right). \tag{11.3}$$

Ćwiczenie 11.1. Wykaż, że jeśli A_{k+1} jest nieosobliwa, to wyrażenie pojawiające się w mianowniku wzoru na s_{k+1} jest różne od zera.

Wskazówka. Zauważ, że $\frac{s_k^T A_k^{-1} F(x_{k+1})}{\|s_k\|_2^2} = v_k^T A_k^{-1} u_k$ *i skorzystaj ze wzoru Shermana–Morrisona.*

Ćwiczenie 11.2. Zaimplementuj metodę Broydena z zastosowaniem wzoru Shermana–Morrisona.

Rozwiązanie. Aby sensownie wyznaczać s_{k+1}, musimy zastanowić się chwilę, jak wyznaczać wektor

$$p_k = A_k^{-1} F(x_{k+1}),$$

występujący we wzorze (11.3). Zauważmy, że na mocy (11.1),

$$A_k^{-1} = \left(\prod_{j=0}^{k-1} \left(I + \frac{s_{j+1} s_j^T}{\|s_j\|_2^2} \right) \right) A_0^{-1},$$

a więc wyznaczenie p_k daje się zrealizować w pętli polegającej na kolejnym mnożeniu przez macierze postaci $\left(I + \frac{s_{j+1} s_j^T}{\|s_j\|_2^2} \right)$. Pamiętajmy, by macierzy postaci $I + uv^T$ nigdy nie formować *explicite*, a w zamian wyznaczać iloczyn $(I + uv^T)p$ zgodnie z wzorem

$$p + u(v^T p),$$

co będzie kosztowało tylko $O(N)$ flopów. Szkielet algorytmu mógłby więc wyglądać następująco:

Bazowa implementacja metody Broydena

$x = x_0$; k = 0;
$A_0 = F'(x)$;
wyznacz rozkład LU lub QR macierzy A_0;
rozwiąż, korzystając z gotowego rozkładu, $A_0 s_0 = -F(x)$;
while not stop **do**
begin
  $x = x + s$;
  oblicz $F(x)$;
  k = k+1;
  rozwiąż, korzystając z gotowego rozkładu, $A_0 s = -F(x)$;
  for j = 0 **to** k−1
  begin
    $s = s + s_{j+1}(s_j^T s)/\|s_j\|_2^2$;
  end
  $s = s/(1 + s_{k-1}^T s/\|s_{k-1}\|_2^2)$;
  $s_k = s$;
end

Jak widać, musimy pamiętać czynniki rozkładu A_0 (samego A_0 już nie) oraz wszystkie pośrednie poprawki $s_0, s_1, \ldots$.

11.2. Zbieżność metody Broydena

Poniższe twierdzenie gwarantuje superliniową zbieżność metody Broydena nawet w przypadku, gdy $A_0 \approx F'(x_0)$.

Twierdzenie 11.1 (o zbieżności metody Broydena)**.** *Przy standardowych założeniach, istnieją δ oraz δ_0 (dostatecznie małe) takie, że jeśli $x_0 \in K(x^*, \delta)$ oraz $\|A_0 - F'(x^*)\| < \delta_0$, to metoda Broydena startująca z x_0 i A_0 jest dobrze określona oraz $x_k \to x^*$ superliniowo.*

Dowód. Zobacz np. [9]. □

Ćwiczenie 11.3. Wykaż, że jeśli $A_0 = F'(x_0)$ oraz x_0 jest dostatecznie blisko x^*, to powyższe twierdzenie zachodzi.

Rozwiązanie. To oczywiste, gdyż

$$\|A_0 - F'(x^*)\| = \|F'(x_0) - F'(x^*)\| \leqslant L\|x_0 - x^*\| \leqslant L\delta.$$

Biorąc δ, δ_0 z twierdzenia o zbieżności metody Broydena i ewentualnie dodatkowo zmniejszając δ tak, by jednocześnie $L\delta < \delta_0$, dostajemy

$$\|A_0 - F'(x^*)\| \leqslant L\delta < \delta_0,$$

co gwarantuje spełnienie wszystkich założeń twierdzenia 11.1.

Ćwiczenie 11.4. Wykaż, że macierze A_k generowane w metodzie Broydena spełniają *równanie siecznych*,

$$A_{k+1}(x_{k+1} - x_k) = F(x_{k+1}) - F(x_k).$$

Rozwiązanie. Z definicji

$$A_{k+1} = A_k + \frac{1}{s_k^T s_k} F(x_{k+1}) s_k^T,$$

więc mnożąc przez $x_{k+1} - x_k = s_k$ mamy

$$A_{k+1}s_k = A_k s_k + F(x_{k+1}) = -F(x_k) + F(x_{k+1}).$$

Ćwiczenie 11.5. Niech $A \in \mathbb{R}^{N\times N}$ oraz $s, y \in \mathbb{R}^N$ przy czym $s \neq 0$. Wtedy

$$\min_{B\in\mathbb{R}^{N\times N}:\, Bs=y} \|A - B\|_2$$

jest osiągane dla

$$B^* = A + \frac{(y - As)s^T}{s^T s}.$$

Znaczy to, że metoda Broydena generuje ciąg macierzy A_k taki, że A_{k+1} jest najbliższe A_k w normie spektralnej wśród wszystkich macierzy spełniających równanie siecznych

$$A_k(x_k - x_{k-1}) = F(x_k) - F(x_{k-1}).$$

Rozwiązanie. Wystarczy pokazać, że dla dowolnej innej macierzy B spełniającej $Bs = y$, norma różnicy $B - A$ nie jest mniejsza niż dla $B^* - A$. Mamy

$$B^* - A = \frac{(y - As)s^T}{s^T s} = \frac{(Bs - As)s^T}{s^T s} = \frac{(B - A)ss^T}{s^T s},$$

skąd

$$\|B^* - A\|_2 \leqslant \|B - A\|_2 \|ss^T\| / \|s\|_2^2 = \|B - A\|_2.$$

Ostatnia równość wynika z faktu, że $\|xy^T\|_2 = \|x\|_2 \, \|y\|_2$.

Na zakończenie wspomnijmy, że — podobnie jak w przypadku modyfikacji metody Newtona — rozsądne jest wykonać co pewną liczbę iteracji *restart* metody Broydena, to znaczy: uruchomić procedurę na nowo, biorąc za punkt początkowy ostatnio wyznaczone przybliżenie. W ten sposóby spowodujemy aktualizację macierzy pochodnej (tej z „zerowego" kroku) oraz zmniejszymy liczbę czynników iloczynu macierzy rzędu jeden koniecznych do obliczenia poprawki.

12. Globalizacja zbieżności

Zarówno metoda Newtona, jak i omawiane przez nas jej modyfikacje, mogą zawieść (to znaczy: mogą nie być zbieżne lub dobrze określone), gdy początkowe przybliżenie x_0 będzie *zbyt daleko* od rozwiązania i założenia standardowe nie będą spełnione w x_0. Ponieważ zwykle trudno wiedzieć *a priori*, czy jesteśmy dostatecznie blisko rozwiązania, opracowano szereg mniej lub bardziej heurystycznych technik *globalizacji* zbieżności metod typu Newtona, pozwalających w końcu dotrzeć w takie pobliże rozwiązania, w którym już nastąpi szybka zbieżność.

Podobny charakter ma inny — jakże często spotykany w zastosowaniach — problem: równanie nieliniowe *z parametrem*, w którym celem jest wyznaczenie nie jednego, jak dotychczas, ale całej *rodziny rozwiązań* (w zależności od parametru). Przykładowo, mogłaby nas interesować charakterystyka prądowo–napięciowa pewnego układu elektronicznego o nieliniowych własnościach: wtedy dla zadanych wartości napięcia musielibyśmy rozwiązać równanie nieliniowe, określające prąd w obwodzie. Ponieważ aby narysować dokładny wykres takiej charakterystyki, należałoby próbkować wartości natężenia w bardzo wielu (powiedzmy, kilkuset) podanych napięciach — dobrze byłoby te kilkaset układów równań rozwiązać możliwie efektywnie. Pokażemy więc pożyteczną metodę wyznaczania bardzo dobrych punktów startowych x_0 w przypadku, gdy zależność rozwiązania od parametru jest dostatecznie regularna.

Co ciekawe, tę technikę można wykorzystać także w celu globalizacji zbieżności metody typu Newtona dla pojedynczego równania nieliniowego!

12.1. Metoda nawrotów

Przypomnijmy, że rozważane przez nas metody iteracyjne są postaci

$$x_{k+1} = x_k - s.$$

Zacznijmy od prostej obserwacji, że — gdy x_k jest daleko od rozwiązania — poprawka $s = F'(x_k)^{-1}F(x_k)$ jest co prawda słuszna co do kierunku (idziemy wszak w stronę, w którą funkcja lokalnie maleje), ale być może przesadna co do wielkości i spowodować, że w następnej iteracji wartość residuum wzrośnie, a nie zmaleje...

Metoda nawrotów (ang. *backtracking*) reprezentuje prostą strategię, polegającą na zachowaniu kierunku poprawki i następnie na ewentualnym jej skróceniu tak, by residuum w x_{k+1} było mniejsze od residuum w x_k: dzięki temu zawsze będziemy redukować $\|F(x)\|$ w nadziei, że tym samym będziemy zbliżać się do rozwiązania.

12.1.1. Prosta metoda nawrotów

W swojej najbardziej podstawowej (raczej nie stosowanej) postaci, metoda nawrotów wyznacza poprawkę s następująco:

Metoda nawrotów, wersja bazowa

$s = F'(x_k)^{-1}F(x_k)$; {*zaczynamy od standardowej poprawki z metody Newtona*}
TEST: $x = x_k - s$; {*wypróbowujemy poprawkę*}

if $\|F(x)\| < \|F(x_k)\|$ **then**
  $x_{k+1} = x$; {*akceptujemy poprawkę*}
else begin
  $s = s/2$; {*skracamy poprawkę i ponownie sprawdzamy*}
  goto TEST;
end

W praktyce powyższe kryterium akceptacji poprawki $\|F(x)\| < \|F(x_k)\|$ może być zbyt łagodne i dlatego raczej stosuje się tzw. regułę Armijo:

Metoda nawrotów z regułą Armijo

$s = F'(x_k)^{-1}F(x_k)$;
$\lambda = 1$; {*mnożnik poprawki*}
TEST: $x = x_k - \lambda s$; {*wypróbowujemy poprawkę*}
if $\|F(x)\| < (1 - \alpha\lambda)\|F(x_k)\|$ **then**
  $x_{k+1} = x$; {*akceptujemy poprawkę*}
else begin
  $\lambda = \lambda/2$; {*zmniejszamy mnożnik i ponownie sprawdzamy*}
  goto TEST;
end

Parametr relaksacyjny α dobiera się na wyczucie, zwykle trochę poniżej jedności. Oczywiście, nie można dać się zwariować i stosować *zbyt małych* mnożników λ, bo prowadziłoby to do stagnacji metody (lepiej potraktować to jako informację o kłopotach ze zbieżnością i następnie restartować metodę z zupełnie innym x_0).

Metoda nawrotów, jakkolwiek często istotnie poprawia charakter zbieżności metody Newtona, może także załamać się, gdy $F'(x_k)$ stanie się osobliwa. Nie musi także gwarantować zbieżności do prawdziwego rozwiązania, gdyż może zdarzyć się, że na przykład
— iteracje x_k dążą do lokalnego ekstremum F, a nie do miejsca zerowego;
— $\|x_k\| \to \infty$ (bo na przykład $F(x) \to 0$ dla $\|x\| \to \infty$).

12.1.2. Wielomianowe nawroty

Zamiast stosować dość prymitywną regułę wyznaczania nowego mnożnika poprawki: $\lambda = \lambda/2$, można wybierać nowe λ jako argument minimalizujący wartość wielomianu interpolacyjnego dla funkcji $g(\lambda) = \|F(x_k - \lambda s)\|$, opartego na węzłach będących poprzednio wybieranymi wartościami λ. Taką strategię nazywa się strategią nawrotów wielomianowych. Inna metoda — obszaru ufności (ang. *trust region*) — pozwala dobrać nie tylko mnożnik poprawki, ale dodatkowo nieco zmienić jej kierunek; jest więc nieco bardziej elastyczna, ale za cenę wyższego kosztu.

Przykład 12.1 (Metoda wielomianowych nawrotów oparta na dwóch punktach)**.** Wybierając

$$g(\lambda) = \|F(x_k - \lambda s)\|_2^2,$$

po teście $\lambda = 1$, dysponujemy następującymi trzema(!) wartościami g:

$$\begin{aligned} g(0) &= \|F(x_k)\|_2^2 \text{ (z poprzedniego kroku)}, \\ g'(0) &= g'(\lambda)_{|_{\lambda=0}} = -2F(x_k)^T F'(x_k)s, \\ g(1) &= \|F(x_k - s)\|_2^2. \end{aligned}$$

Zauważmy, że jeśli s było wyznaczone metodą Newtona, to wtedy faktycznie $g'(0) < 0$, a więc $F(x_k)$ maleje w kierunku s. Na podstawie tych trzech wartości możemy skonstruować

wielomian interpolacyjny (Hermite'a) drugiego stopnia oraz wyznaczyć λ_{nowe}, minimalizujące ten wielomian:

$$\lambda_{\text{nowe}} = -\frac{1}{2}\frac{g'(0)}{g(1) - g(0) - g'(0)}.$$

Ćwiczenie 12.1. Wyprowadź powyższy wzór na λ_{nowe}.

Wskazówka. Wspominany wielomian interpolacyjny Hermite'a dla g to

$$w(\lambda) = g(0) + g'(0)\lambda + (g(1) - g(0) - g'(0))\lambda^2.$$

Dla metody Broydena także można sformułować regułę podobną do reguły Armijo dla metody Newtona, [9].

Jeszcze innym sposobem globalizacji zbieżności metody Newtona może być wykorzystanie metody kontynuacji — o czym w następnym rozdziale.

12.2. Metody kontynuacji

W wielu praktycznych zastosowaniach mamy do czynienia nie z jednym równaniem nieliniowym, ale z całą *rodziną* równań, indeksowaną pewnym parametrem (lub zestawem parametrów). Dla ustalenia uwagi, rozważmy więc rodzinę równań indeksowaną parametrem $t \in [0, 1]$,

$$H(x, t) = 0, \tag{12.1}$$

gdzie $H : D \times [0, 1] \to \mathbb{R}^N$, gdzie D jest otwartym niepustym podzbiorem $\mathbb{R}^N$.

W ogólności zbiór wszystkich rozwiązań (12.1) może być dziwaczny, ale my zajmiemy się regularnym — spotykanym w niektórych zastosowaniach — przypadkiem, kiedy możemy lokalnie jednoznacznie określić funkcję $t \mapsto x(t)$ taką, że

$$H(x(t), t) = 0.$$

Dokładniej, załóżmy, że $H \in C^1(D \times [0, 1])$ i dla pewnego parametru $t^* \in (0, 1)$ i dla pewnego $x^* \in D$ zachodzi

$$H(x^*, t^*) = 0.$$

Jeśli dodatkowo założymy, że pochodna cząstkowa $H_x(x^*, t^*)$ jest nieosobliwa, to na mocy twierdzenia o funkcji uwikłanej istnieje $U = (t^* - \epsilon, t^* + \epsilon)$, gdzie $\epsilon > 0$, takie, że dla $t \in U$ jest jednoznacznie wyznaczone rozwiązanie $x(t)$,

$$H(x(t), t) = 0.$$

Ponadto, $x \in C^1(U)$ oraz

$$x'(t) = -H_x(x(t), t)^{-1} H_t(x(t), t) \qquad \text{dla } t \in U. \tag{12.2}$$

Mówimy w takim przypadku, że $x(t)$ jest *regularną gałęzią* rozwiązań (12.1), przechodzącą przez (x^*, t^*). W dalszym ciągu zajmiemy się właśnie metodami wyznaczania takiej regularnej gałęzi rozwiązań.

12.2.1. Wyznaczanie dobrego przybliżenia początkowego dla metody typu Newtona

Przypuśćmy, że dla parametru t^* wyznaczyliśmy już rozwiązanie $x^* = x(t^*)$. Naszym zadaniem jest wyznaczenie $x(t^*+h)$, gdzie $h > 0$ jest małe[1]. Odpowiada to wziętej z życia sytuacji, gdy chcemy prześledzić przebieg gałęzi $x(t)$, próbkując jej wartości dla kolejnych (zapewne gęsto rozmieszczonych w U) wartości parametru t.

Naturalnie, do wyznaczenia $x(t^*+h)$ możemy użyć dowolnej z wcześniej opisywanych metod — ale czy nie udałoby się wykorzystać informacji uzyskanej dla t^* w celu wyznaczenia przybliżenia x_0 dla $x(t^*+h)$ tak, by metoda Newtona od razu startowała z *dobrego* przybliżenia początkowego? Ze względu na regularny charakter gałęzi $x(t)$ nietrudno zgadnąć, że niezłym przybliżeniem dla $x(t^*+h)$ mogłoby być $x_0 = x^*$. Rzeczywiście, niech

$$l = \max_{t\in[t^*,t^*+h]} \|x'(t)\|.$$

Wtedy (na mocy lematu o wartości średniej 9.1)

$$\|x(t^*+h) - x^*\| \leqslant l|h|,$$

więc jeśli h jest dostatecznie małe względem l, to x_0 będzie dostatecznie blisko $x(t^*+h)$, by zagwarantować szybką (kwadratową) zbieżność metody Newtona (pod oczywistym warunkiem, że spełnione są założenia standardowe, por. rozdział 9.3.1). Ten sposób wyboru x_0, zaproponowany przez Poincaré'go, nazywa się *klasyczną metodą kontynuacji*.

Można jednak, niewiele większym kosztem, uzyskać jeszcze lepsze przybliżenie początkowe dla metody Newtona (lub podobnej).

Ponieważ zachodzi (12.2) z warunkiem początkowym $x(t^*) = x^*$, to przybliżoną wartość $x(t^*+h)$ możnaby uzyskać, stosując jakiś schemat różnicowy dla (12.2). Na przykład, korzystając dla (12.2) z jawnego schematu Eulera z krokiem h, jako $x_0 \approx x(t^*+h)$ położymy

$$x_0 = x(t^*) + h\,x'(t^*) = x^* - h\cdot H_x(x^*,t^*)^{-1}H_t(x^*,t^*).$$

Ten sposób wyznaczania początkowego przybliżenia $x(t^*+h)$ nazywa się *kontynuacją wzdłuż stycznej*. Aby więc wyznaczyć x_0, musimy rozwiązać układ równań z macierzą pochodnej $H_x(x^*,t^*)$ (wyznaczoną dla poprzednio znalezionego rozwiązania — to przecież macierz występująca w iteracji Newtona!) no i wyznaczyć wartość $H_t(x^*,t^*)$ — co zwykle nie jest zbyt trudne analitycznie (w przeciwnym przypadku, możemy zadowolić się, jak zwykle, różnicową aproksymacją).

Jeśli $x \in C^2(U)$, to jakość tak uzyskanego przybliżenia jest znacznie wyższa niż w przypadku klasycznej metody kontynuacji. Rzeczywiście,

$$x(t^*+h) - x_0 = x(t^*+h) - x(t^*) - h\,x'(t^*) = \int_0^1 h\,\left(x'(t^*+\tau h) - x'(t^*)\right)\,d\tau,$$

skąd, oznaczając $\tilde{l} = \max_{t\in[t^*,t^*+h]} \|x''(t)\|$, dostajemy na mocy lematu o wartości średniej oszacowanie

$$\|x(t^*+h) - x_0\| \leqslant \int_0^1 |h|\,\|x'(t^*+\tau h) - x'(t^*)\|\,d\tau \leqslant \frac{\tilde{l}}{2}h^2.$$

Ćwiczenie 12.2. Dlaczego w metodzie kontynuacji wzdłuż stycznej, zastosowanie schematu niejawnego do rozwiązywania (12.2) nie miałoby większego sensu?

[1] W szczególności, $t^*+h \in U$.

Rozwiązanie. Bo, po pierwsze, wymagałoby rozwiązania podobnego równania nieliniowego, a po drugie — byłoby niepotrzebnie kosztowne. Przecież nas i tak interesuje tylko wykonanie *jednego* kroku takiego schematu!

W świetle powyższego, nabiera znaczenia kwestia doboru właściwej wielkości kroku kontynuacji h. Na razie dysponujemy jedynie informacją jakościową, że „dla h dostatecznie małego", dostaniemy zbieżność. Jednak w praktyce chcielibyśmy używać *rozsądnie małych* (czytaj: nie za małych) h — bo obniża to koszt wyznaczenia aproksymacji $x(t)$ dla $t \in U$. Istnieją na szczęście strategie *kontroli kroku* kontynuacji h [4], analogiczne jak w przypadku równań różniczkowych, ale dostosowane do tej konkretnej sytuacji: zagwarantowania (kwadratowej) zbieżności metody Newtona startującej z x_0.

Warto także wiedzieć, że istnieją bardziej wyrafinowane techniki kontynuacji, które pozwalają przejść przez punkty krytyczne gałęzi (w których pochodna po t jest nieokreślona)

12.2.2. Metoda homotopii

Jeśli trudno znaleźć dobry punkt startowy dla metody Newtona dla równania

$$F(x) = 0,$$

możemy spróbować *sztucznie* wprowadzić parametr $t \in [0, 1]$ i określić równanie (12.1) tak, by

$$H(x, 1) = F(x)$$

oraz

$$H(x, 0) = \tilde{F}(x),$$

przy czym dla $\tilde{F}$ jesteśmy w stanie łatwo wygenerować dobry punkt startowy. Jeśli tak, to postępując zgodnie z metodą kontynuacji — rozwiązując $H(x, t_i) = 0$ dla kolejnych wartości t_i, poczynając od zera — dojdziemy w końcu do $t_i = 1$, i tym samym wygenerujemy dlań dobre przybliżenie startowe.

Taka metoda globalizacji zbieżności metody Newtona — przez wykorzystanie kontynuacji dla odpowiednio spreparowanej rodziny zadań z parametrem — nosi nazwę metody homotopii, gdyż w sposób ciągły przeprowadzamy zadanie „łatwe" na zadanie „trudne". Należy jednak pamiętać, że w praktyce ta metoda może, ale nie musi się sprawdzić, gdyż znalezienie *dobrej* homotopii H może być sprawą znacznie trudniejszą niż znalezienie dobrego x_0! Zazwyczaj „naturalne" pomysły na H, np. wziąć „bardzo łatwą" $\tilde{F}(x)$ i określić

$$H(x, t) = tF(x) + (1 - t)\tilde{F}(x),$$

albo położyć jeszcze prostszą

$$H(x, t) = tF(x_0) + (F(x) - F(x_0)) \quad \text{dla zadanego } x_0,$$

są *zbyt* proste, aby były skuteczne.

13. Kwadratury interpolacyjne w wielu wymiarach

13.1. Sformułowanie zadania

Ostatnie trzy wykłady poświęcimy numerycznemu całkowaniu funkcji wielu zmiennych. Dokładniej, dla danej funkcji $f : [0,1]^d \to \mathbb{R}$ chcemy obliczyć (przybliżyć) wartość

$$I_d(f) \;=\; \int_{[0,1]^d} f(\vec{x})\, d\vec{x} \;=\; \underbrace{\int_0^1 \int_0^1 \cdots \int_0^1}_{d} f(x_1, x_2, \ldots, x_d)\, dx_1 dx_2 \cdots dx_d.$$

Zakładamy, że powyższa całka istnieje. W ogólniejszym sformułowaniu, chcielibyśmy obliczyć całkę z wagą ω funkcji $f : \mathbb{R}^d \to \mathbb{R}$, która jest postaci

$$I_{d,\omega}(f) \;=\; \int_{\mathbb{R}^d} f(\vec{x})\, \omega(\vec{x})\, d\vec{x}.$$

Waga ω jest tutaj nieujemna i całkowalna.

Zauważmy, że ograniczenie się w ostatnim przypadku do $\mathbb{R}^d$ nie zmniejsza ogólności, gdyż całkę po dowolnym mierzalnym obszarze $D \subseteq \mathbb{R}^d$ można wymodelować przyjmując, że waga $\omega(\vec{x}) = 0$ dla wszystkich $\vec{x} \notin D$.

Zadanie całkowania funkcji wielu zmiennych ma ogromne znaczenie praktyczne i dlatego warto znać skuteczne metody numeryczne jego rozwiązywania.

Przykład 13.1. Wycena obecnej wartości wielu instrumentów finansowych, w tym tzw. opcji, opiera się na założeniu, że przyszłe ceny podlegają losowym zmianom kolejnych odcinkach czasowych. Obecna wartość opcji obliczana jest jako wartość oczekiwana funkcji wypłaty. Odpowiada to obliczaniu całki oznaczonej funkcji d zmiennych, gdzie d jest liczbą odcinków czasowych. Jest to często całka ze standardową d wymiarową wagą gaussowską postaci

$$(2\pi)^{-d/2} \int_{\mathbb{R}^d} f(\xi_1, \ldots, \xi_d) \exp\Big(-\frac{1}{2}(\xi_1^2 + \cdots, \xi_d^2)\Big)\, d\xi_1 \cdots d\xi_d,$$

przy czym f jest (zwykle skomplikowaną) funkcją wypłaty na końcu okresu, a ξ_j reprezentują czynniki losowe w kolejnych odcinkach czasu. Wymiar d może wynosić nawet kilka tysięcy.

Z podstawowego wykładu z metod numerycznych każdy z nas wie jak numerycznie całkować funkcje jednej zmiennej. Stosowane metody w znakomitej większości przypadków sprowadzają się do scałkowania funkcji, która jest kawałkami wielomianem określonego stopnia interpolującym funkcję podcałkową. Pomysł ten może być uogólniony na przypadek funkcji wielu zmiennych. Aby jednak mówić o kwadraturach interpolacyjnych w wielu wymiarach, musimy najpierw zastanowić się nad rozwiązywalnością odpowiedniego zadania interpolacyjnego.

13.2. Interpolacja na siatkach regularnych

13.2.1. Postać wielomianu interpolacyjnego

Niech

$$a \leqslant t_1 < t_2 < \cdots < t_r \leqslant b.$$

Jeśli f jest funkcją jednej zmiennej, $f : [a, b] \to \mathbb{R}$, to wielomian

$$p_f(x) = \sum_{j=1}^{r} f(t_j)\, l_j(x),$$

gdzie l_j jest j-tym wielomianem Lagrange'a,

$$l_j(x) = \prod_{j \neq i=1}^{r} \frac{x - t_j}{t_i - t_j}, \qquad 1 \leqslant j \leqslant r,$$

(przy czym $l_1 \equiv 1$ dla $r = 1$) jest stopnia co najwyżej $(r-1)$ i interpoluje f w punktach t_j, tzn. przyjmuje w tych punktach te same wartości co f. W przypadku $d \geqslant 2$ możemy podobnie zdefiniować „wielowymiarowe" wielomiany Lagrange'a.

W tym celu zakładamy, że na każdej współrzędnej dany jest przedział, a w nim układ r punktów

$$a^{(k)} \leqslant t_1^{(k)} < t_2^{(k)} < \ldots < t_r^{(k)} \leqslant b^{(k)}, \qquad 1 \leqslant k \leqslant d.$$

Oznaczając przez $l_j^{(k)}$ odpowiednie wielomiany Lagrange'a jednej zmiennej dla k-tego podziału, definiujemy wielomiany Lagrange'a d zmiennych jako

$$l_{j_1,\ldots,j_d}(x_1, \ldots, x_d) = l_{j_1}^{(1)}(x_1)\, l_{j_2}^{(2)}(x_2) \cdots l_{j_d}^{(d)}(x_d)$$

dla wszystkich $1 \leqslant j_k \leqslant r$, $1 \leqslant k \leqslant d$. Dla skrócenia zapisu, będziemy dalej używać zapisu wektorowego $\vec{j} = (j_1, \ldots, j_d)$, a $1 \leqslant \vec{j} \leqslant d$ będzie oznaczać, że nierówności zachodzą dla każdej współrzędnej j_k, $1 \leqslant k \leqslant d$. Podobnie, $t_{\vec{j}} = (t_{j_1}^{(1)}, t_{j_2}^{(2)}, \ldots, t_{j_d}^{(d)})$.

Wielomiany $l_{\vec{j}}$ należą do przestrzeni $\mathcal{P}_d^r$ wielomianów d zmiennych postaci

$$p(\vec{x}) = p(x_1, \ldots, x_d) = \sum_{0 \leqslant \vec{i} \leqslant r-1} a_{\vec{i}} \cdot x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d},$$

gdzie $a_{\vec{i}}$ są dowolnymi wsoółczynnikami rzeczywistymi. Zauważmy, że $p \in \mathcal{P}_d^r$ wtedy i tylko wtedy gdy p jest wielomianem stopnia co najwyżej $(r-1)$ ze względu na każdą zmienną x_k.

Lemat 13.1. *Jeśli wielomian $p \in \mathcal{P}_d^r$ zeruje się we wszystkich r^d punktach $t_{\vec{j}}$, $1 \leqslant \vec{j} \leqslant r$, to p jest wielomianem zerowym.*

Dowód. Dowód przeprowadzimy przez indukcję ze względu na wymiar d. Dla $d = 1$ lemat jest oczywiście prawdziwy, bo na podstawie zasadniczego twierdzenia algebry niezerowy wielomian stopnia co najwyżej $(r-1)$ nie może mieć r różnych zer.

Niech $d \geqslant 2$. Niech $a_{\vec{j}}$ będą współczynnikami wielomianu p. Dla ustalonej k zdefiniujmy wielomian $p_k \in \mathcal{P}_{d-1}^r$ jako

$$p_k(x_1, \ldots, x_{d-1}) = p(x_1, \ldots, x_{d-1}, t_k^{(d)}).$$

Wielomian ten zeruje się w $r(d-1)$ punktach $t_{i_1,\ldots,i_{d-1}}$. Zapisując go w postaci

$$p_k(x_1, \ldots, x_{d-1}) = \sum_{0 \leqslant i_1,\ldots,i_{d-1} \leqslant d-1} b_{i_1,\ldots,i_{d-1}}^{(k)} \cdot x_1^{i_1} \cdots x_{d-1}^{i_{d-1}},$$

gdzie współczynniki

$$b^{(k)}_{i_1,\ldots,i_{d-1}} = \sum_{0\leqslant i_d\leqslant d-1} a_{\vec{i}} \cdot (t^{(d)}_k)^{i_d},$$

oraz stosując założenie indukcyjne mamy, że $b^{(k)}_{i_1,\ldots,i_{d-1}} = 0$. A więc dla wszystkich wyborów indeksów $i_1,\ldots,i_{d-1}$ wielomian jednej zmiennej $\sum_{i_d=0}^{d-1} a_{\vec{i}} \cdot t^{i_d}$ zeruje się w r punktach $t = t^{(d)}_s$. To zaś wymusza $a_{\vec{i}} = 0$ dla wszystkich $0 \leqslant \vec{i} \leqslant d-1$ i w konsekwencji $p \equiv 0$. □

Lemat 13.1 wykorzystamy do pokazania następującego twierdzenia.

Twierdzenie 13.1. *Wielomiany $l_{\vec{j}}$, $1 \leqslant \vec{j} \leqslant r$, tworzą bazę przestrzeni $\mathcal{P}^r_d$. W szczególności,* $\dim(\mathcal{P}^r_d) = r^d$.

Dowód. Zauważmy, że podobnie jak w przypadku $d = 1$,

$$l_{\vec{j}}(t_{\vec{i}}) = \begin{cases} 1, & \text{jeśli } \vec{i} = \vec{j}, \\ 0, & \text{w przeciwnym przypadku.} \end{cases}$$

Stąd, jeśli kombinacja liniowa $p = \sum_{\vec{j}} \alpha_{\vec{j}} l_{\vec{j}}$ jest wielomianem zerowym to dla wszystkich $\vec{i}$

$$0 = p(t_{\vec{i}}) = \sum_{\vec{j}} \alpha_{\vec{j}} l_{\vec{j}}(t_{\vec{i}}) = \alpha_{\vec{i}},$$

czyli układ $\{l_{\vec{i}} : 1 \leqslant \vec{i} \leqslant r\}$ jest liniowo niezależny. Z drugiej strony, układ ten rozpina $\mathcal{P}^r_d$, bo dla dowolnego wielomianu p z tej przestrzeni mamy

$$p = \sum_{1\leqslant\vec{j}\leqslant r} p(t_{\vec{j}})\, l_{\vec{j}}. \tag{13.1}$$

Rzeczywiście, w przeciwnym przypadku różnica wielomianu p i prawej strony (13.1) byłaby niezerowym wielomianem w $\mathcal{P}^r_d$, który zeruje się we wszystkich r^d punktach $t_{\vec{i}}$. To zaś przczyłoby lematowi 13.1. □

Stąd już jeden krok do następującego wniosku podsumowującego nasze dotychczasowe rozważania. Niech

$$D = [a^{(1)}, b^{(1)}] \times [a^{(2)}, b^{(2)}] \times \cdots \times [a^{(d)}, b^{(d)}]$$

będzie d wymiarowym prostokątem.

Wniosek 13.1. *Dla dowolnej funkcji $f : D \to \mathbb{R}$ wielomian*

$$p_f(\vec{x}) = \sum_{0\leqslant\vec{j}\leqslant r} f(t_{\vec{j}})\, l_{\vec{j}}(\vec{j})$$

jest jedynym wielomianem w $\mathcal{P}^r_d$ interpolującym f w punktach $t_{\vec{j}}$ tzn. takim, że

$$p_f(t_{\vec{j}}) = f(t_{\vec{j}})$$

dla wszystkich $1 \leqslant \vec{j} \leqslant r$.

13.2.2. Błąd interpolacji

Zastanówmy się teraz jaki jest błąd otrzymanej interpolacji. Dla uproszczenia będziemy od teraz zakładać, że D jest kostką d wymiarową, tzn. wszystkie krawędzie mają tą samą długość, którą oznaczymy przez H, a węzły na każdej współrzędnej

$$t_j^{(k)} = a^{(k)} + u_j H, \qquad 1 \leqslant j \leqslant r,$$

gdzie

$$0 \leqslant u_1 < u_2 < \cdots < u_r \leqslant 1$$

jest pewną ustaloną siatką na odcinku jednostkowym.

W przypadku skalarnym, o ile funkcja f jest r-krotnie różniczkowalna w sposób ciągły, to

$$f(x) - p_f(x) \;=\; (x-t_1)(x-t_2)\cdots(x-t_r)\frac{f^{(r)}(\xi)}{r!},$$

przy czym $\xi \in [a,b]$ zależy od x. Stąd w szczególności mamy

$$|f(x) - p_f(x)| \;\leqslant\; \frac{(b-a)^r}{r!}\|f^{(r)}\|_\infty, \tag{13.2}$$

gdzie $\|f^{(r)}\|_\infty = \max_{a\leqslant t\leqslant b}|f^{(r)}(t)|$. Aby wyprowadzić formułę na błąd interpolacji w przypadku wielowymiarowym, będziemy potrzebować pewnego prostego uogólnienia ostatniego wzoru.

Załóżmy, że zamiast dokładnych wartości $f(t_i)$ mamy jedynie wartości przybliżone y_i takie, że błąd

$$|y_i - f(t_i)| \;\leqslant\; \delta, \qquad 1 \leqslant i \leqslant r. \tag{13.3}$$

Niech dalej $\tilde{p}_f$ będzie wielomianem stopnia co najwyżej $(r-1)$ interpolującym dane przybliżone y_i w punktach t_i. Ponieważ $(p_f - \tilde{p}_f)$ jest wielomianem interpolującym dane $f(t_j) - y_j$, na podstawie wzoru (13.1) mamy

$$|p_f(x) - \tilde{p}_f(x)| \;\leqslant\; \delta \cdot \sum_{i=1}^{r} |l_i(x)| \;\leqslant\; \delta \cdot S_r,$$

gdzie $S_1 = 1$, a dla $r \geqslant 2$

$$S_r = \max_{0\leqslant z\leqslant 1} \sum_{i=1}^{r} \prod_{i\neq j=1}^{r} \left|\frac{z-u_j}{u_i-u_j}\right|.$$

Stąd i z formuły na błąd interpolacji dla dokładnych danych otrzymujemy

$$|f(x) - \tilde{p}_f(x)| \;\leqslant\; |f(x) - p_f(x)| + |p_f(x) - \tilde{p}_f(x)| \;\leqslant\; \frac{(b-a)^r}{r!}\|f^{(r)}\|_\infty \;+\; \delta \cdot S_r. \tag{13.4}$$

Wprowadzimy jeszcze klasę $\mathcal{F}_r(D)$ funkcji $f : D \to \mathbb{R}$, które w całej swojej dziedzinie są r-krotnie różniczkowalne w sposób ciągły ze względu na każdą zmienną. Dla $f \in \mathcal{F}_r(D)$ definiujemy

$$B_r(f) \;=\; \max_{1\leqslant i\leqslant d} \left\{ \left\| \frac{\partial^r f}{\partial x_1^r} \right\|_\infty, \ldots, \left\| \frac{\partial^r f}{\partial x_d^r} \right\|_\infty \right\}.$$

Twierdzenie 13.2. *Niech* $D = [a^{(1)}, a^{(1)} + H] \times \cdots \times [a^{(d)}, a^{(d)} + H]$. *Jeśli* $f \in \mathcal{F}_r(D)$ *to dla każdego* $\vec{x} \in D$ *błąd interpolacji*

$$|f(\vec{x}) - p_f(\vec{x})| \leqslant \frac{H^r}{r!} C_{r,d} B_r(f),$$

gdzie $C_{1,d} = d$, *a dla* $r \geqslant 2$

$$C_{r,d} = \frac{S_r^d - 1}{S_r - 1}.$$

Dowód. Rozpatrzymy tylko $r \geqslant 2$ pozostawiając przypadek $r = 1$ jako proste ćwiczenie.

Dla $d = 1$ nierówność w tezie jest równoważna (13.2). Załóżmy więc, że $d \geqslant 2$. Ponieważ dla każdego ustalonego $t_k^{(d)}$ wielomian $(d-1)$ zmiennych $p_f(x_1, \ldots, x_{d-1}, t_k^{(d)})$ jest wielomianem interpolacyjnym dla funkcji $(d-1)$ zmiennych $f(x_1, \ldots, x_{d-1}, t_k^{(d)})$, na podstawie założenia indukcyjnego mamy

$$|f(x_1, \ldots, x_{d-1}, t_k^{(d)}) - p_f(x_1, \ldots, x_{d-1}, t_k^{(d)})| \leqslant \frac{H^r}{r!} B_r(f) \left(\frac{S_r^{d-1} - 1}{S_r - 1} \right). \tag{13.5}$$

Zauważmy, że dla ustalonych z kolei pierwszych $(d-1)$ współrzędnych $x_1, \ldots, x_{d-1}$ wielomian $p_f(x_1, \ldots, x_{d-1}, t)$ jest wielomianem jednej zmiennej t interpoluącym funkcję jednej zmiennej $f(x_1, \ldots, x_{d-1}, t)$ w punktach $t_k^{(d)}$ na podstawie danych zaburzonych na poziomie δ równym prawej stronie (13.5). Stąd i z (13.4) ostatecznie otrzymujemy

$$\begin{aligned} |f(\vec{x}) - p_f(\vec{x})| &\leqslant \frac{H^r}{r!} B_r(f) + \delta \cdot S_r \\ &= \frac{H^r}{r!} B_r(f) \left(1 + \frac{S_r^{d-1} - 1}{S_r - 1} S_r \right) \\ &= \frac{H^r}{r!} B_r(f) \left(\frac{S_r^d - 1}{S_r - 1} \right). \end{aligned}$$

□

13.3. Kwadratury interpolacyjne

13.3.1. Kwadratury proste

Jesteśmy już dobrze uzbrojeni w mechanizm interpolacyjny i możemy zdefiniować wielowymiarowe kwadratury interpolacyjne dla całkowania funkcji $f : D \to \mathbb{R}$ zdefiniowanych na kostce

$$D = [a^{(1)}, a^{(1)} + H] \times \cdots \times [a^{(d)}, a^{(d)} + H].$$

Kwadratury te dane są równością

$$Q_{r,d}(f) = \int_D p_f(\vec{x})\, d\vec{x}, \tag{13.6}$$

gdzie $p_f \in \mathcal{P}_d^r$ jest wielomianem interpolującym f w punktach $t_{\vec{j}}$, $1 \leqslant \vec{j} \leqslant r$.

Chociaż postać (13.6) kwadratury znakomicie nadaje się do rozważań teoretycznych, nie jest jednak praktyczna ze względu na obliczenia. Zauważmy, że

$$\begin{aligned} Q_{r,d}(f) &= \int_D \sum_{\vec{j}} f(t_{\vec{j}}) l_{\vec{j}}(\vec{x})\, d\vec{x} = \sum_{\vec{j}} f(t_{\vec{j}}) \int_D l_{\vec{j}}(\vec{x})\, d\vec{x} \\ &= H^d \cdot \sum_{\vec{j}} f(t_{\vec{j}}) \prod_{k=1}^{d} \left(\int_0^1 l_{j_k}(u)\, du \right), \end{aligned}$$

gdzie l_j jest j-tym wielomianem Lagrange'a dla punktów $u_1, u_2, \ldots, u_d$. Stąd, wprowadzając oznaczenie

$$a_k = \int_0^1 l_k(u)\, du,$$

kwadraturę interpolacyjną można zapisać w postaci

$$Q_{r,d}(f) = H^d \cdot \sum_{1 \leqslant j_1, \ldots, j_d \leqslant r} a_{j_1} a_{j_2} \cdots a_{j_d} \cdot f(t_{j_1}^{(1)}, t_{j_2}^{(2)}, \ldots, t_{j_d}^{(d)}).$$

Zauważmy, że a_k są współczynnikami jednowymiarowej kwadratury interpolacyjnej $Q_r(f) = \sum_{k=1}^{r} a_k f(t_k)$ opartej na punktach u_k, przybliżającej całkę $\int_0^1 f(u)\, du$. Mówiąc inaczej, zdefiniowana przez nas wielowymiarowa kwadratura interpolacyjna jest d-produktem tensorowym wybranej kwadratury jednowymiarowej.

Na koniec tego podrozdziału podamy oszacowanie błędu kwadratury $Q_{r,d}$. Ponieważ

$$\int_D f(\vec{x})\, d\vec{x} - Q_{r,d}(f) = \int_D \left(f(\vec{x}) - p_f(\vec{x}) \right) d\vec{x},$$

z twierdzenia 13.2 natychmiast otrzymujemy następujący wniosek.

Wniosek 13.2. *Jeśli $f \in \mathcal{F}_r(D)$ to błąd kwadratury interpolacyjnej $Q_{r,d}$ jest ograniczony przez*

$$\left| \int_D f(\vec{x})\, d\vec{x} - Q_{r,d}(f) \right| \leqslant \frac{H^{r+d}}{r!} C_{r,d}\, B_r(f).$$

13.3.2. Kwadratury złożone

Podobnie jak w przypadku funkcji jednej zmiennej, definiujemy kwadratury złożone dla funkcji wielu zmiennych. Dla uproszczenia zakładamy, że całkujemy po kostce jednostkowej $[0,1]^d$.

Dla danego n wprowadzamy podział kostki na n^d podkostek

$$\left[\frac{i_1 - 1}{n}, \frac{i_1}{n} \right] \times \left[\frac{i_2 - 1}{n}, \frac{i_2}{n} \right] \times \cdots \left[\frac{i_d - 1}{n}, \frac{i_d}{n} \right], \qquad 1 \leqslant i_k \leqslant n, \quad 1 \leqslant k \leqslant d.$$

Następnie na każdej podkostce stosujemy prostą kwadraturę interpolacyjną opartą na siatce regularnej składającej się z r^d punktów. Skonstruowaną w ten sposób kwadraturę złożoną oznaczymy przez $Q_{r,d}^{(n)}$.

Przykład 13.2. Jeśli bazową kwadraturą jednowymiarową jest reguła punktu środkowego,

$$Q_1(f) = (b-a) \cdot f\left(\frac{a+b}{2} \right),$$

to wynikową kwadraturą złożoną na $[0,1]^d$ jest po prostu reguła prostokątów

$$Q_{r,d}^{(n)}(f) = \left(\frac{1}{n} \right)^d \cdot \sum_{1 \leqslant i_1, \ldots, i_d \leqslant n} f\left(\frac{i_1 - 1/2}{n}, \cdots, \frac{i_d - 1/2}{n} \right).$$

Nasze rozważania wieńczy twierdzenie o błędzie kwadratury złożonej, które natychmiast wynika z wniosku 13.2 oraz sposobu konstrukcji kwadratury.

Twierdzenie 13.3. *Kwadratura złożona $Q_{r,d}^{(n)}$ korzysta z co najwyżej*

$$N = (r\,n)^d$$

wartości funkcji f. Jeśli $f \in \mathcal{F}_r([0,1]^d)$ to jej błąd

$$\left| \int_{[0,1]^d} f(\vec{x})\,d\vec{x} - Q_{r,d}^{(n)}(f) \right| \leqslant \left(\frac{1}{N}\right)^{r/d} \left(\frac{r^r}{r!}\right) C_{r,d}\,B_r(f).$$

13.4. Przekleństwo wymiaru

Złożone kwadratury interpolacyjne mogą być z powodzeniem stosowane dla niskich wymiarów, powiedzmy $d = 2, 3$. Dla dużych wymiarów d mają one bowiem tą niepożądaną własność, że liczba węzłów rośnie wykładniczo szybko wraz z zagęszczaniem siatki. Nawet jeśli weźmiemy po 2 punkty na każdej współrzędnej to całkowita liczba punktów siatki regularnej wyniesie 2^d. Pamiętamy, że w wielu praktycznych zastosowaniach d może sięgać nawet kilku tysięcy. W takich przypadkach obliczenie wartości kwadratury jest zadaniem praktycznie niewykonalnym.

To jednak nie koniec złych wiadomości. Przyjrzyjmy się jeszcze błędowi złożonej kwadratury interpolacyjnej. Twierdzenie 13.3 mówi, że błąd ten jest ograniczony z góry proporcjonalnie do $N^{-r/d}$, gdzie N jest liczbą wszystkich użytych punktów. To drugi powód do niepokoju, uzasadniony poniższym przykładem.

Przykład 13.3. Załóżmy, że chcemy całkować funkcję 360 zmiennych i jako kwadraturę bazową stosujemy kwadraturę Simpsona, dla której $r = 4$. Górne ograniczenie błędu sugeruje, że aby być pewnym wyniku z dokładnością 10^{-2} to musimy obliczyć wartości funkcji w aż 10^{180} punktach. Czy naprawdę jest aż tak źle?

Rzeczywiście jest tak źle, a nawet gorzej. Okazuje się, że rzędu zbieżości $N^{-r/d}$ nie da się poprawić w klasie funkcji $\mathcal{F}_r([0,1]^d)$. Mówi o tym następujące twierdzenie.

Twierdzenie 13.4. *Istnieje $c = c_{r,d} > 0$ o następującej własności. Dla dowolnej aproksymacji całki wykorzystującej N wartości funkcji można znaleźć funkcję $f \in \mathcal{F}_r([0,1]^d)$ dla której $B_r(f) = 1$, a błąd aproksymacji całki wynosi co najmniej $c\,N^{-r/d}$.*

Dowód. Załóżmy, że dana aproksymacja całki oblicza wartości funkcji w punktach $\vec{t}_j$, $1 \leqslant j \leqslant N$. Dowód twierdzenia polega na konstrukcji dwóch funkcji, f_+ i f_-, które zerują się we wszystkich $\vec{t}_j$ (a tym samym ich całki są aproksymowane tą samą liczbą), dla których $B_r(f_+) = 1 = B_r(f_-)$, ale różnica całek

$$\int_{[0,1]^d} (f_+ - f_-)(\vec{t})\,d\vec{t} \geqslant 2c\,N^{-r/d},$$

dla pewnej c niezależnej od f i d. Wtedy, przynajmniej dla jednej z tych funkcji błąd aproksymacji całki wynosi co najmniej $cN^{-r/d}$.

Wybierzmy n taką, że $n^d > N$ i skonstruujmy na $[0,1]^d$ regularną siatkę składającą się z n^d kostek, każda o krawędzi długości $h = 1/n$.

Niech dalej $\phi : \mathbb{R} \to \mathbb{R}$ będzie dowolną funkcją r-krotnie różniczkowalną w sposób ciągły spełniającą następujące warunki:

1. $\phi(x) = 0$ dla $x \notin (0,1)$,
2. $\phi^{(j)}(0) = 0 = \phi^{(j)}(1)$ dla $0 \leqslant j \leqslant r$,
3. $\int_0^1 \phi(t)\,dt =: a > 0$.

Każdej kostce

$$K_{\vec{i}} := [(i_1 - 1)h, i_1 h] \times \cdots \times [(i_d - 1)h, i_d h]$$

naszej regularnej siatki przyporządkujemy funkcję

$$\phi_{\vec{i}}(x_1, \ldots, x_d) := h^r \prod_{k=1}^{d} \phi(x_k/h - i_k).$$

Zauważmy, że $B_r(\phi_{\vec{i}}) = 1$ oraz

$$\int_{K_{\vec{i}}} \phi_{\vec{i}}(\vec{t})\,d\vec{t} = a^d\, h^{r+d}.$$

Jasne jest, że istnieje co najmniej $n^d - N$ multi-indeksów $\vec{i}$ (kostek) takich, że żaden z punktów $\vec{t}_j$ nie należy do wnętrza $K_{\vec{i}}$. Oznaczmy zbiór takich indeksów przez S i zdefiniujmy funkcje

$$f_+ := \sum_{\vec{i} \in S} \phi_{\vec{i}}, \qquad f_- := -f_+.$$

Wtedy obie funkcje zerują się w $\vec{t}_j$, $B_r(f_+) = 1 = B_r(f_-)$, oraz

$$\int_{[0,1]^d} f_+(\vec{t})\,d\vec{t} = -\int_{[0,1]^d} f_-(\vec{t})\,d\vec{t} \geqslant (n^d - N)\, a^d\, h^{r+d}.$$

Podstawiając $n = \lceil N^{1/d}(1 + d/r)^{1/d} \rceil$ dostajemy ostatecznie

$$\int_{[0,1]^d} f_+(\vec{t})\,d\vec{t} \geqslant c\, N^{-r/d}, \qquad \text{gdzie} \quad c = \frac{d a^d}{r 2^{r+d} (1 + d/r)^{1+r/d}}.$$

□

Opisane zjawisko nosi nazwę *przekleństwa wymiaru*.

14. Metody Monte Carlo

14.1. Wstęp, metody niedeterministyczne

Poprzedni wykład zakończyliśmy pesymistycznym twierdzeniem 13.4, że nie istnieją efektywne metody numerycznego całkowania funkcji wielu zmiennych, ponieważ ma miejsce zjawisko przekleństwa wymiaru. Zwróćmy jednak uwagę na to, że fakt istnienia przekleństwa wymiaru stwierdziliśmy przy założeniach, że:
(i) model obliczeniowy jest *deterministyczny*,
(ii) funkcje podcałkowe są r-krotnie różniczkowalne po każdej zmiennej.
Można mieć nadzieję, że przekleństwo wymiaru zniknie, albo zostanie złagodzone, gdy przynajmniej jedno z tych założeń nie będzie spełnione.

Ten wykład poświęcimy (klasycznej) metodzie Monte Carlo numerycznego całkowania, która jest przykładem metody niedeterministycznej, tzn. takiej, która oblicza wynik wykorzystując zjawiska losowe. Chociaż może to brzmieć dziwnie, to właśnie niedeterministyczne zachowanie metody pozwala pokonać przekleństwo wymiaru.

Opisana dalej klasyczna metoda Monte Carlo związana jest ściśle ze Stanisławem Ulamem, uczniem Stefana Banacha i reprezentantem Lwowskiej Szkoły Matematycznej. Ulam zastosował metodę Monte Carlo do obliczania skomplikowanych całek w ramach „Manhattan Project" w Los Alamos (USA), w czasie II Wojny światowej.

14.2. Klasyczna metoda Monte Carlo

14.2.1. Definicja i błąd

Tak jak w poprzednim rozdziale chcemy obliczyć całkę

$$I_d(f) := \int_{[0,1]^d} f(\vec{x})\, d\vec{x} = \underbrace{\int_0^1 \int_0^1 \cdots \int_0^1}_{d} f(x_1, x_2, \ldots, x_d)\, dx_1 dx_2 \cdots dx_d.$$

Zakładamy przy tym, że $f : [0,1]^d \to \mathbb{R}$ jest funkcją, której kwadrat jest całkowalny,

$$\int_{[0,1]^d} |f(\vec{x})|^2\, d\vec{x} < \infty.$$

Definicja 14.1. *Klasyczna metoda Monte Carlo* polega na przybliżeniu $I_d(f)$ średnią arytmetyczną wartości funkcji f w losowo wybranych punktach, tzn.

$$MC_{d,N}(f) = MC_{d,N}(f; \vec{t}_1, \vec{t}_2, \ldots, \vec{t}_N) := \frac{1}{N} \cdot \sum_{j=1}^{N} f(\vec{t}_j),$$

gdzie $\vec{t}_1, \vec{t}_2, \ldots, \vec{t}_N$ są punktami wylosowanymi niezależnie od siebie, każdy zgodnie z rozkładem jednostajnym na $[0,1]^d$.

Konsekwencją zastosowania losowości jest to, że przy różnych realizacjach metody otrzymujemy różne wyniki, w zależności od wyboru punktów $\vec{t}_j$. Wynik $MC_{d,N}(f)$ jest więc zmienną losową, której wartość oczekiwana wynosi

$$\begin{aligned} E\left(MC_{d,N}(f)\right) &= \int_{[0,1]^{d\cdot N}} MC_{d,N}(f;\vec{t}_1,\ldots,\vec{t}_N)\, d\vec{t}_1\cdots d\vec{t}_N \\ &= \frac{1}{N}\sum_{j=1}^{N}\int_{[0,1]^d} f(\vec{t})\, d\vec{t} = I_d(f). \end{aligned}$$

Ponieważ różnica $I_d(f) - MC_{d,N}(f)$ jest też zmienną losową, za błąd metody Monte Carlo dla danej funkcji f przyjmiemy odchylenie standardowe,

$$e(f;MC_{d,N}) := \sqrt{E\left(I_d(f) - MC_{d,N}(f)\right)^2}.$$

Twierdzenie 14.1. *Dla danej funkcji f błąd metody Monte Carlo wynosi*

$$e(f;MC_{d,N}) = \frac{\sigma(f)}{\sqrt{N}},$$

gdzie

$$\sigma(f) := \sqrt{I_d(f^2) - I_d^2(f)}$$

jest wariancją funkcji f.

Zanim przystąpimy do dowodu zauważmy, że $\sigma(f)$ jest dobrze zdefiniowaną wielkością, bowiem nierówność

$$|I_d(f)| \leqslant \sqrt{I_d(f^2)}$$

jest szczególnym przypadkiem znanej *nierówności Schwarza* dla całek.

Dowód. Oznaczmy, dla uproszczenia, zmienną losową $X = MC_{d,N}(f)$. Wtedy

$$E(X - E(X))^2 = E\left(X(X - E(X)) - E(X)(X - E(X))\right) = E(X^2) - E^2(X). \tag{14.1}$$

Ponadto

$$\begin{aligned} E(X^2) &= E\left(\frac{1}{N}\left(\sum_{j=1}^{N} f(\vec{t}_j)\right)^2\right) = \frac{1}{N^2}E\left(\sum_{j=1}^{N} f^2(\vec{t}_j) + \sum_{i\neq j} f(\vec{t}_i)f(\vec{t}_j)\right) \\ &= \frac{1}{N^2}\left(NI_d(f^2) + (N^2-N)I_d^2(f)\right) = \frac{1}{N}I_d(f^2) + \left(1 - \frac{1}{N}\right)I_d^2(f), \end{aligned}$$

gdzie skorzystaliśmy z niezależności zmiennych losowych $f(t_i)$ i $f(t_j)$ dla $i \neq j$. Stąd i z (14.1) dostajemy

$$e^2(f;MC_{d,N}) = E(X - E(X))^2 = \frac{1}{N}I_d(f^2) + \left(1 - \frac{1}{N}\right)I_d^2(f) - I_d^2(f) = \frac{1}{N}\left(I_d(f^2) - I_d^2(f)\right),$$

co kończy dowód. □

Uwaga 14.1. Zauważmy, że w dowodzie pokazaliśmy przy okazji nierówność Schwarza posługując się narzędziami rachunku prawdopodobieństwa.

Twierdzenie (14.1) mówi, że błąd metody Monte Carlo jest proporcjonalny do $N^{-1/2}$ przy bardzo słabych wstępnych założeniach na funkcję (jedynie całkowalność kwadratu funkcji). Jest to istotna poprawa w porównaniu do błędu $N^{-r/d}$ dla metod deterministycznych. W szczególności ważne jest, że wykładnik $1/2$ przy N^{-1} jest niezależny od wymiaru d, a konsekwencją tego pokonanie przekleństwa wymiaru.

Dziwnym może wydawać się, że przekleństwo wymiaru można zlikwidować używając metod niedeterministycznych (losowych). Jednak niczego nie ma za darmo. Należy pamiętać, że jest to możliwe za cenę niepewności wyniku. O ile bowiem metoda deterministyczna produkuje zawsze ten sam wynik, metoda niedeterministyczna (taka jak Monte Carlo) produkuje różne wyniki zależnie od konkretnych realizacji zmiennych losowych. Dlatego, mimo iż błąd oczekiwany jest proporcjonalny do $N^{-1/2}$ to nie mamy całkowitej pewności, że przy konkretnej realizacji otrzymany wynik jest tego samego rzędu. Z tego punktu widzenia warto przytoczyć następującą równość, która wynika z *centralnego twierdzenia granicznego*; mianowicie, dla dowolnych $c_1 < c_2$ mamy

$$\lim_{N\to\infty} \mathrm{Prob}\left(\frac{c_1\sigma(f)}{\sqrt{N}} \leqslant I_d(f) - MC_{d,N}(f) \leqslant \frac{c_2\sigma(f)}{\sqrt{N}}\right) = \frac{1}{\sqrt{2\pi}}\int_{c_1}^{c_2} e^{-t^2/2}dt,$$

gdzie Prob oznacza prawdopodobieństwo względem rozkładu jednostajnego na $[0,1]^{dN}$.

14.2.2. Całkowanie z wagą

Deterministyczne metody interpolacyjne z poprzedniego rozdziału można stosować jedynie do całkowania na d-wymiarowych prostokątach. Metoda Monte Carlo ma oprócz wymienionych również i tą zaletę, że łatwo ją uogólnić na przypadek całkowania z wagą. Dla przybliżenia wartości

$$I_{d,\omega}(f) := \int_{\mathbb{R}^d} f(\vec{x})\omega(\vec{x})\,d\vec{x}, \qquad \text{gdzie} \qquad \int_{\mathbb{R}^d} \omega(\vec{x})\,d\vec{x} =: W < \infty,$$

możemy bowiem zastosować wzór

$$MC^{\omega}_{d,N}(f) := \frac{W}{N}\cdot\sum_{j=1}^{N} f(\vec{t}_j),$$

przy czym $\vec{t}_1,\ldots,\vec{t}_N$ są tym razem punktami wybranymi losowo i niezależnie od siebie, zgodnie z rozkładem na $\mathbb{R}^d$ o gęstości ω/W.

Adaptując odpowiednio dowód twierdzenia 14.1 otrzymujemy następujące wyrażenie na błąd uogólnionej metody Monte Carlo.

Twierdzenie 14.2. *Niech* $I_{d,\omega}(f^2) = \int_{\mathbb{R}^d} f^2(\vec{x})\omega(\vec{x})\,d\vec{x} < \infty$. *Wtedy*

$$e(f; MC^{\omega}_{d,N}) = \frac{\sigma_\omega(f)}{\sqrt{N}},$$

gdzie

$$\sigma_\omega(f) = \sqrt{W\,I_{d,\omega}(f^2) - I^2_{d,\omega}(f)}.$$

14.3. Redukcja wariancji

Zauważyliśmy, że zaletą metody Monte Carlo jest nie tylko jej prostota, ale również to, że błąd średni wynosi $\sigma_\omega(f)N^{-1/2}$. Naturalnym jest teraz pytanie, czy błędu tego nie można poprawić. Temu celowi służą metody *redukcji wariancji*, które polegają w ogólności na redukcji czynnika $\sigma_\omega(f)$. Spośród wielu technik redukcji wariancji skupimy uwagę na dwóch: losowaniu warstwowemu oraz funkcjach kontrolnych. Dla uproszczenia będziemy zakładać, że całkujemy z wagą jednostkową na kostce

$$D = [0,1]^d.$$

14.3.1. Losowanie warstwowe

Podzielmy obszar całkowania D na K rozłącznych podzbiorów D_i tak, że

$$D = \bigcup_{i=1}^{K} D_i$$

i zastosujmy Monte Carlo do całkowania po każdym D_i, tzn. całkę $\int_D f(\vec{x})\,d\vec{x}$ przybliżymy wielkością

$$\overline{MC}_{d,N}(f) := \sum_{i=1}^{K} MC^{(i)}_{d,N_i}(f),$$

gdzie $MC^{(i)}_{d,N_i}$ jest metodą Monte Carlo zastosowaną do całki

$$I^{(i)}_d(f) := \int_{D_i} f(\vec{x})\,d\vec{x}, \qquad 1 \leqslant i \leqslant K,$$

oraz $N = \sum_{i=1}^{K} N_i$.

Oznaczmy przez $|D_i|$ objętość d-wymiarową podzbioru D_i. Ponieważ zmienne losowe $I^{(i)}_d(f) - MC^{(i)}_{d,N_i}(f)$ są parami niezależne dla $1 \leqslant i \leqslant K$, na podstawie twierdzenia 14.2 mamy

$$\begin{aligned} E\left((I_d(f) - \overline{MC}_{d,N}(f))^2\right) &= E\left(\left(\sum_{i=1}^{K} I^{(i)}_d(f) - MC_{d,N_i}(f)\right)^2\right) \\ &= \sum_{i=1}^{K} E\left((I^{(i)}_d(f) - MC^{(i)}_{d,N_i}(f))^2\right) \\ &= \sum_{i=1}^{K} \frac{1}{N_i}\left(|D_i|\, I^{(i)}_d(f^2) - (I^{(i)}_d(f))^2\right). \end{aligned}$$

Przyjmijmy teraz, że

$$N_i = |D_i| \cdot N, \qquad 1 \leqslant i \leqslant K,$$

przy czym dla uproszczenia (ale bez utraty ogólności) zakładamy, że wielkości te są całkowite. Wtedy otrzymujemy

$$e(f, \overline{MC}_{d,N}) = \frac{1}{\sqrt{N}} \cdot \sqrt{I_d(f^2) - \sum_{i=1}^{K} \frac{1}{|D_i|}(I^{(i)}_d(f))^2}. \tag{14.2}$$

Błąd tak zdefiniowanej metody $\overline{MC}_{d,N}$ nie jest większy od błędu klasycznej metody $MC_{d,N}$ z Twierdzenia 14.1.

Twierdzenie 14.3. *Dla dowolnej funkcji f takiej, że $I_d(f^2) < \infty$ mamy*

$$e(f, \overline{MC}_{d,N}) \leqslant e(f, MC_{d,N}),$$

przy czym równość zachodzi tylko wtedy gdy iloraz $I_d^{(i)}(f)/|D_i|$ jest stały, niezależnie od i.

Dowód. Rzeczywiście, oznaczając

$$a_i = \sqrt{|D_i|}, \qquad b_i = \frac{I_d^{(i)}(f)}{\sqrt{|D_i|}},$$

oraz wykorzystując nierówność Schwarza dla ciągów mamy

$$I_d^2(f) = \left(\sum_{i=1}^K a_i b_i\right)^2 \leqslant \left(\sum_{i=1}^K a_i^2\right)\left(\sum_{i=1}^K b_i^2\right) = \sum_{i=1}^K b_i^2 = \sum_{i=1}^K \frac{1}{|D_i|}(I_d^{(i)}(f))^2,$$

przy czym równość zachodzi tylko wtedy gdy wektory $(a_1, \ldots, a_K)^T$ i $(b_1, \ldots, b_K)^T$ są liniowo zależne, co jest równoważne warunkowi w treści twierdzenia.

Prawdziwość tezy pokazuje teraz porównanie wzorów na błędy obu metod. □

Widzimy, że stosując losowanie warstwowe z ustalonym podziałem na K podzbiory D_i możemy co prawda zmiejszyć błąd, ale szybkość zbieżności $N^{-1/2}$ pozostaje ta sama. A czy można poprawić zbieżność stosując różne podziały dla różnych wartości N? Okazuje się, że tak, o ile założymy pewną regularność funkcji f.

Aby to uzyskać, najpierw przekształcimy wzór (14.2) na błąd metody $\overline{MC}_{d,N}$ do postaci

$$e(f, \overline{MC}_{d,N}) = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^K I_d^{(i)}((f - c_i)^2)}. \tag{14.3}$$

gdzie

$$c_i := \frac{I_d^{(i)}(f)}{|D_i|}, \qquad 1 \leqslant i \leqslant K.$$

Załóżmy teraz, że f spełnia warunek Lipschitza ze stałą L,

$$|f(\vec{x}) - f(\vec{y})| \leqslant L \cdot \|\vec{x} - \vec{y}\|_\infty, \qquad \vec{x}, \vec{y} \in D.$$

Wtedy istnieją $\vec{x}_i \in \overline{D}_i$ takie, że $f(\vec{x}_i) = c_i$, a stąd i z lipschitzowskości f mamy, że dla dowolnego $\vec{x} \in D_i$

$$|f(\vec{x}) - c_i| \leqslant L \cdot \|\vec{x} - \vec{x}_i\|_\infty \leqslant L \cdot \operatorname{diam}_\infty(D_i),$$

gdzie

$$\operatorname{diam}_\infty(D_i) := \sup\{\|\vec{x} - \vec{y}\|_\infty : \vec{x}, \vec{y} \in D_i\}$$

jest średnicą zbioru D_i w normie max. W konsekwencji, ze wzoru (14.3) dostajemy następujące oszacowanie błędu:

$$e(f, \overline{MC}_{d,N}) \leqslant \frac{L}{\sqrt{N}}\sqrt{\sum_{i=1}^K |D_i| \operatorname{diam}^2(D_i)}.$$

Ustalmy teraz równomierny podział kostki D na $K = N$ podkostek D_i, każda o krawędzi długości $N^{-1/d}$ (zakładamy, bez zmniejszenia ogólności, że $N^{1/d}$ jest całkowita) tak, że nasza metoda aproksymuje całkę na D_i używając tylko jednej wartości. Wtedy $\mathrm{diam}(D_i) = N^{-1/d}$ oraz

$$e(f, \overline{MC}_{d,N}) \leqslant L \cdot N^{-(1/2+1/d)}.$$

Ostatecznie, otrzymany w ten sposób wariant losowania warstwowego jest zbieżny z wykładnikiem większym niż $1/2$. Oczywiście, może to mieć praktyczne znaczenie jedynie dla małych wymiarów d, bowiem dla dużych d wykładnik $1/2 + 1/d$ jest właściwie równy $1/2$.

14.3.2. Funkcje kontrolne

Podobny efekt zwiększenia szybkości zbieżności można uzyskać stosując klasyczną Monte Carlo bezpośrednio do funkcji $f - g$, gdzie g jest pewną specjalnie dobraną funkcją, zwaną *funkcją kontrolną*.

Rzeczywiście, przedstawmy f w postaci $f = g + (f - g)$. Wtedy

$$I_d(f) = I_d(g) + I_d(f - g),$$

co sugeruje zastosowanie następującej metody:

$$\widetilde{MC}_{d,N}(f) := I_d(g) + MC_{d,N}(f - g).$$

Ponieważ

$$\begin{aligned} I_d(f) - \widetilde{MC}_{d,N}(f) &= (I_d(g) + I_d(f - g)) - (I_d(g) + MC_{d,N}(f - g)) \\ &= I_d(f - g) - MC_{d,N}(f - g), \end{aligned}$$

z twierdzenia 14.1 natychmiast wynika, że błąd

$$e(f; \widetilde{MC}_{d,N}) = e(f - g; MC_{d,N}) = \frac{\sigma(f - g)}{\sqrt{N}}.$$

Pozostaje kwestia doboru funkcji g tak, aby istotnie zmniejszyć wariancję $\sigma(f - g)$. Weźmy najpierw funkcję schodkową postaci

$$g(\vec{x}) = \sum_{i=1}^{K} c_i \mathbf{1}_{D_i}(\vec{x}),$$

gdzie

$$\mathbf{1}_{D_i}(\vec{x}) = \begin{cases} 1 & \vec{x} \in D_i, \\ 0 & \vec{x} \notin D_i, \end{cases}, \qquad 1 \leqslant i \leqslant K,$$

jest funkcją charakterystyczną zbioru D_i, a $c_i = f(\vec{x}_i)$ dla dowolnych $\vec{x}_i \in D_i$. Oczywiście, najlepiej byłoby wziąć $\vec{x}_i$ tak, aby $c_i = I_d^{(i)}(f)/|D_i|$, ale jest to niemożliwe, bo nie znamy całek $I_d^{(i)}(f)$. (Znajomość tych całek nie była konieczna w przypadku losowania warstwowego!) Wtedy dostajemy ten sam efekt jak dla $\overline{MC}_{d,N}$, tzn. dla lipschitzowskiej f

$$\sigma^2(f - g) \leqslant I_d((f - g)^2) \leqslant C^2 \cdot \sum_{i=1}^{K} |D_i| \,\mathrm{diam}^2(D_i) \tag{14.4}$$

i dla g odpowiadającej równomiernemu podziałowi na N podkostek otrzymujemy błąd proporcjonalny do $N^{-(1/2+1/d)}$.

W ogólności, funkcję g należy w praktyce wybierać tak, aby dobrze aproksymowała funkcję f. Oczywiście, wybór ten musi bazować na informacji jaką posiadamy o f.

Na przykład, dla funkcji r-gładkich, tzn. dla $f \in \mathcal{F}_r(D)$ można w ten sposób dostać jeszcze lepszy wykładnik. Rzeczywiście, niech $g = g_f$ będzie kawałkami wielomianem stopnia najwyżej $r-1$ po każdej zmiennej interpolującym f, dla równomiernego podziału kostki jednostkowej na $N^{1/d}$ podkostek. Z rozdziału 13 wiemy, że wtedy dla funkcji $f \in \mathcal{F}_r(D)$ błąd interpolacji jest postaci (zob. twierdzenie 13.2)

$$|f(\vec{x}) - g_f(\vec{x})| \leqslant \frac{C_{r,d} B_r(f)}{r!} \cdot N^{-r/d}$$

W konsekwencji, dla tak skonstruowanej metody dostajemy następujący błąd oczekiwany.

Twierdzenie 14.4. *Dla $f \in \mathcal{F}_r(D)$ mamy*

$$e(f; \widetilde{MC}_{d,N}) \leqslant \frac{C_{r,d} B_r(f)}{r!} \cdot N^{-(1/2+r/d)}.$$

Dodajmy, że $\widetilde{MC}_{d,N}$ jest w istocie metodą mieszaną, gdyż używa wartości funkcji f obliczanych w N punktach wybranych deterministycznie oraz w N punktach wybranych losowo.

Zbieżności $N^{-(1/2+r/d)}$ nie da się już poprawić w klasie $\mathcal{F}_r(D)$. Dokładniej, można pokazać następujące twierdzenie, które jest odpowiednikiem twierdzenia 13.4 dla algorytmów niedeterministycznych.

Twierdzenie 14.5. *Istnieje $c > 0$ o następującej własności: dla dowolnej (deterministycznej lub niedeterministycznej) aproksymacji całki wykorzystującej N wartości funkcji istnieje $f \in \mathcal{F}_r([0,1]^d)$ dla której $B_r(f) = 1$, a błąd oczekiwany aproksymacji całki wynosi co najmniej $c\, N^{-(1/2+r/d)}$.*

14.4. Generowanie liczb (pseudo-)losowych

Dotychczas milcząco przyjmowaliśmy, że umiemy generować ciągi

$$X_1, X_2, X_3, \ldots, X_n, \ldots$$

niezależnych liczb losowych zgodnie z danym rozkładem prawdopodobieństwa. Nie jest to jednak zadanie trywialne. W praktyce obliczeniowej liczby losowe uzyskujemy przez zastosowanie specjalnych programów. Ponieważ komputer jest urządzeniem deterministycznym, tak uzyskane ciągi nie są idealnie losowe już choćby dlatego, że są okresowe. Fakt ten wpływa na pogorszenie jakości wyniku i w szczególności powoduje, że ich użycie pozwala uzyskać jedynie kilka liczb znaczących, przy czym im większy wymiar d zadania tym gorsza graniczna dokładność. Z tych względów mówimy raczej o *generatorach liczb pseudo-losowych.*

Generowanie liczb pseudolosowych jest bardzo obszernym tematem, my tylko zwrócimy uwagę na podstawowe metody.

14.4.1. Liniowy generator kongruencyjny

Liniowe generatory kongruencyjne służą generowaniu ciągów losowych U_i o rozkładzie jednostajnym na odcinku $[0,1]$ i zdefiniowane są w następujący prosty sposób. Startujemy z $U_0 = x_0$ i kolejno obliczamy dla $i = 1, 2, 3, \ldots$

$$\left\{ \begin{array}{lcl} x_i & := & (a\, x_{i-1} + c) \bmod m, \\ U_i & := & x_i/m, \end{array} \right. .$$

Jakość takiego generatora zależy istotnie o wyboru liczb całkowitych a, b i m. W szczgólności pożądane jest, aby generator miał maksymalny okres m. Jeśli $c \neq 0$ to warunkami dostatecznymi na to są:
(a) c i m są względnie pierwsze,
(b) jeśli p dzieli m to p dzieli $a - 1$,
(c) jeśli 4 dzieli m to 4 dzieli $a - 1$.

Dostęp do dobrego generatora liczb losowych o rozkładnie jednostajnym $U \sim \mathrm{Unif}([0,1])$ jest ważny również z tego względu, że jest on zwykle podstawą dla konstrukcji generatorów ciągów o bardziej skomplikowanych rozkładach prawdopodobieństwa.

14.4.2. Odwracanie dystrybuanty i „akceptuj albo odrzuć"

Jeśli znana jest dystrybuanta żądanego rozkładu, czyli funkcja

$$F(x) := \mathrm{Prob}(X \leqslant x),$$

oraz łatwo obliczyć jej odwrotność zdefiniowaną jako

$$F^{-1}(u) := \inf\{x : F(x) \geqslant u\},$$

to potrzebne ciągi losowe mogą być wygenerowane według wzoru

$$X := F^{-1}(U), \qquad U \sim \mathrm{Unif}([0,1]).$$

Rzeczywiście, mamy

$$\mathrm{Prob}(X \leqslant x) = \mathrm{Prob}(F^{-1}(U) \leqslant x) = \mathrm{Prob}(U \leqslant F(x)) = F(x).$$

Na przykład, jeśli $F(x) = 1 - e^{-x^2/2}$ to można zastosować wzór $X = \sqrt{-2\ln(U)}$.

Niestety, dla wielu rozkładów dystrybuanta nie może być dokładnie obliczona. Wtedy jakość metody zależy od jakości zastosowanej numerycznej aproksymacji funkcji $F^{-1}(x)$.

Inna uniwersalna metoda generowania liczb losowych o dowolnym rozkładzie na $\mathbb{R}$, zwana *akceptuj albo odrzuć*, polega na wykorzystaniu istniejącego „dobrego" generatora liczb innego rozkładu na $\mathbb{R}$. Dokładniej, załóżmy, że dysponujemy generatorem liczb losowych Y zgodnie z rozkładem o gęstości g, a interesują nas liczby X pochodzące z rozkładu o gęstości f. Załóżmy ponadto, że

$$f(x) \leqslant c \cdot g(x)$$

dla pewnej stałej $c > 0$. Wtedy możemy użyć następującego algorytmu:

```
{   repeat
        generuj Y zgodnie z g;
        generuj U ~ Unif([0,1])
    until U ⩽ f(Y)/(cg(Y));
    return X := Y
}
```

Aby pokazać poprawność takiego generatora zauważmy, że

$$\begin{aligned}\mathrm{Prob}(X \in A) &= \mathrm{Prob}(\, Y \in A \,:\, U \leqslant f(Y)/(cg(Y))\,) \\ &= \frac{\mathrm{Prob}(\, Y \in A,\, U \leqslant f(Y)/(cg(Y))\,)}{P(\, U \leqslant f(Y)/(cg(Y))\,)}.\end{aligned}$$

Ponieważ dla $a \in [0,1]$ prawdopodobieństwo, że $U \leqslant a$ wynosi a to

$$\mathrm{Prob}(U \leqslant f(Y)/(cg(Y))\,) = \int_{\mathbb{R}} \frac{f(x)}{cg(x)} g(x)\,dx = \frac{1}{c}$$

i w konsekwencji dostajemy

$$\begin{aligned}\mathrm{Prob}(X \in A) &= c \cdot \mathrm{Prob}(Y \in A,\, U \leqslant f(Y)/(cg(Y))\,) \\ &= c \cdot \int_A \frac{f(x)}{cg(x)} g(x)\,dx = \int_A f(x)\,dx.\end{aligned}$$

14.4.3. Metoda Box-Muller dla rozkładu gaussowskiego

Normalny rozkład gaussowski na $\mathbb{R}$ o funkcji gęstości

$$g(t) = \frac{1}{\sqrt{2\pi}}\, e^{-t^2/2}$$

jest najczęściej stosowanym rozkładem niejednostajnym. Dla rozkładu gaussowskiego bardzo efektywne okazują się algorytmy bazujące na odwracaniu dystrybuanty. Używają one dość skomplikowanych aproksymacji funkcji F^{-1}.

Prostszą i najbardziej popularną jest metoda *Box-Muller*. Generuje ona od razu dwie niezależne liczby Z_1 i Z_2 (albo, równoważnie, punkt $(Z_1, Z_2) \in \mathbb{R}^2$ zgodnie z rozkładem normalnym w $\mathbb{R}^2$), na podstawie dwóch liczb losowych o rozkładzie jednostajnym.

Przedstawmy $(x, y) \in \mathbb{R}^2$ we współrzędnych biegunowych,

$$\begin{cases} x &= r \cdot \cos\varphi, \\ y &= r \cdot \sin\varphi, \end{cases}$$

gdzie $r \in [0, \infty)$ i $\varphi \in [0, 2\pi)$. Metoda polega na wygenerowaniu zmiennych φ i r, a następnie zastosowaniu powyższego wzoru. Generowanie φ jest proste, bo ma rozkład jednostajny. Policzmy dystrybuantę rokładu zmiennej r. Mamy

$$\begin{aligned}\mathrm{Prob}(r \leqslant R) &= \frac{1}{2\pi} \int_{x^2+y^2 \leqslant R^2} e^{-(x^2+y^2)/2}\, d(x,y) = \frac{1}{2\pi} \int_0^{2\pi} \int_0^R r e^{-r^2/2} dr d\varphi \\ &= \int_0^R r e^{-r^2/2} dr = -e^{-r^2/2}\Big|_0^R = 1 - e^{-R^2/2}.\end{aligned}$$

Stąd, stosując metodę odwracania dystrybuanty mamy $R = \sqrt{-2\ln U}$, $U \sim \mathrm{Unif}([0,1])$. Rachunki te prowadzą do następującego generatora.

{ generuj $U_1, U_2 \sim \mathrm{Unif}([0,1])$;
$R := -2\ln U_1$; $V := 2\pi U_2$;
$Z_1 := \sqrt{R}\cos(V)$; $Z_2 := \sqrt{R}\sin(V)$;
return Z_1, Z_2
}

15. Metody quasi-Monte Carlo

15.1. Co to są metody quasi-Monte Carlo?

Metody Monte Carlo potrafią pokonać przekleństwo wymiaru, mają jednak również swoje wady. Najważniejsze z nich to:
(i) niepewność wyniku (który ma charakter probablistyczny),
(ii) stosunkowo wolna zbieżność (praktycznie $N^{-1/2}$),
(iii) konieczność stosowania (niekiedy skomplikowanych) generatorów liczb losowych.

Można powiedzieć, że skoro używamy z powodzeniem generatorów liczb pseudo-losowych, to implementacje Monte Carlo są w istocie deterministyczne. Dlatego, wybierając punkty deterministycznie, ale tak, aby w jakiś sposób „udawały" i „uśredniały" wybór losowy, powinniśmy dostać metodę deterministyczną o zbieżności co najmniej $N^{-1/2}$. A przy okazji usunęlibyśmy dwie z wymienionych wad Monte Carlo.

Takie intuicyjne myślenie wydaje się nie mieć racjonalnych podstaw, bo przecież metody deterministyczne podlegają przekleństwu wymiaru. Pamiętajmy jednak, że twierdzenie 13.4 o przekleństwie zachodzi w klasie $\mathcal{F}_r(D)$ funkcji różniczkowalnych r razy po każdej zmiennej. Zwróciliśmy na to uwagę już na początku rozdziału 14. Dlatego zasadne jest poszukiwanie pozytywnych rozwiązań dla funkcji o innej regularności.

Quasi-Monte Carlo jest deterministycznym odpowiednikiem klasycznej metody Monte Carlo dla aproksymacji całek na kostkach,

$$I_d(f) := \int_D f(\vec{x})\, d\vec{x}, \qquad D = [0,1]^d.$$

Definicja 15.1. Metoda *quasi-Monte Carlo* polega na przybliżeniu $I_d(f)$ średnią arytmetyczną,

$$\mathrm{QMC}_{d,N}(f) = \mathrm{QMC}_{d,N}(\vec{t}_1, \ldots, \vec{t}_N) := \frac{1}{n} \cdot \sum_{j=1}^{N} f(\vec{t}_j),$$

gdzie $\vec{t}_1, \vec{t}_2, \ldots, \vec{t}_N$ są pewnymi szczególnymi punktami w D wybranymi w sposób deterministyczny.

Przez długi czas od swojego powstania uważano, że metody quasi-Monte Carlo są efektywne jedynie dla całek o niskich wymiarach. Dopiero pod koniec lat 90-tych ubiegłego wieku zauważono, że dają istotnie lepsze wyniki niż Monte Carlo w obliczaniu wartości niektórych instrumentów finansowych. Obecnie quasi-Monte Carlo jest powszchnie uznaną i bardzo popularną metodą, której znaczenie trudno przecenić mimo, że dotychczas nie udało się znaleźć pełnego teoretycznego wytłumaczenia jej efektywności.

15.2. Dyskrepancja

Rozważania na temat quasi-Monte Carlo zaczniemy od zdefiniowania pojęcia *dyskrepancji*, które odgrywa fundamentalną rolę w analizie efektywności tych metod.

Dla $\vec{x} = [x_1, \ldots, x_d]^T \in D$ niech

$$[\vec{0}, \vec{x}) \;:=\; [0, x_1) \times \cdots \times [0, x_d)$$

oznacza d wymiarowy prostokąt w D, „zakotwiczony" w zerze. Zakładamy przy tym, że $[\vec{0}, \vec{0}) = \emptyset$.

Definicja 15.2. Dyskrepancją (z gwiazdką) danego zbioru N punktów $\vec{t}_j \in [0,1)^d$, $1 \leqslant j \leqslant N$, nazywamy wielkość

$$\mathrm{DISC}_d^*(\vec{t}_1, \ldots, \vec{t}_N) \;=\; \sup_{\vec{x} \in D} \Big| \mathrm{DISC}_d(\vec{x}; \vec{t}_1, \ldots, \vec{t}_N) \Big|,$$

gdzie

$$\mathrm{DISC}_d(\vec{x}; \vec{t}_1, \ldots, \vec{t}_N) \;=\; x_1 \ldots x_d - \frac{\#\left\{ j \;:\; \vec{t}_j \in [\vec{0}, \vec{x}) \right\}}{N},$$

a $\#A$ oznacza liczbę elementów zbioru A.

Ponieważ $x_1 \cdots x_d$ jest d-wymiarową objętością prostokąta $[\vec{0}, \vec{x})$, dyskrepancja pokazuje jak dobrze danych N punktów aproksymuje objętości tych prostokątów. Równoważnie, oznaczając przez $\mathbf{1}_{[\vec{0},\vec{x})}$ funkcję charakterystyczną prostokąta $[\vec{0}, \vec{x})$ mamy

$$\int_D \mathbf{1}_{[\vec{0},\vec{x})}(t)\, dt = x_1 \cdots x_d \qquad \text{oraz} \qquad \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}_{[\vec{0},\vec{x})}(\vec{t}_j) = \frac{\#\{j \;:\; \vec{t}_j \in [\vec{0}, \vec{x})\}}{N}.$$

Stąd dyskrepancję można również interpretować jako maksymalny błąd aproksymacji funkcji charakterystycznych prostokątów przez odpowiedni algorytm quasi-Monte Carlo.

Pojęcie dyskrepancji zilustrujemy najpierw na przykładzie jednowymiarowym $d = 1$. Nietrudno pokazać, że dla dowolnych t_j

$$\mathrm{DISC}_1^*(t_1, \ldots, t_N) \;\geqslant\; \frac{1}{2N}. \tag{15.1}$$

Rzeczywiście, $\mathrm{DISC}_1(x; t_1, \ldots, t_N)$ jako funkcja x ma na każdym z przedziałów $[0, t_1)$, $[t_{j-1}, t_j)$, $2 \leqslant j \leqslant N$, pochodną równą 1, oraz przyjmuje zero dla $x = 0$. Zatem, jeśli dyskrepancja byłaby mniejsza od $1/(2N)$ to mielibyśmy

$$t_1 < \frac{1}{2N}, \qquad t_j - t_{j-1} < \frac{1}{N}, \quad 2 \leqslant j \leqslant N,$$

a stąd

$$t_N \;=\; t_1 + \sum_{j=2}^{N} (t_j - t_{j-1}) \;<\; \frac{1}{2N} + \frac{N-1}{N} \;=\; 1 - \frac{1}{2N}.$$

Otrzymujemy sprzeczność, bo dla $t_N < x < 1 - 1/(2N)$

$$\mathrm{DISC}_1(x; t_1, \ldots, t_N) \;<\; \left(1 - \frac{1}{2N}\right) - 1 \;=\; -\frac{1}{2N}.$$

Z dowodu wynika, że równość w (15.1) zachodzi jedynie dla równomiernego rozmieszczenia punktów,

$$t_j^* \;=\; \frac{j - 1/2}{N}, \qquad 1 \leqslant j \leqslant N.$$

W tym przypadku algorytm $\mathrm{QMC}_{1,N}$ redukuje się do zasady punktu środkowego.

Z punktu widzenia praktycznych obliczeń dobrze byłoby mieć ciąg nieskończony

$$t_1, t_2, \ldots, t_n, \ldots \subset [0,1)$$

i konstruować kolejne aproksymacje używając N początkowych wyrazów tego ciągu. Ciekawe, że wtedy zbieżność N^{-1} nie może być zachowana. Dokładniej, można pokazać istnienie $c > 0$ takiego, że dla każdego ciągu nierówność

$$\mathrm{DISC}_1^*(t_1, \ldots, t_N) \geqslant c\,\frac{\ln N}{N}$$

zachodzi dla nieskończenie wielu N.

Analiza dyskrepancji w wymiarach $d \geqslant 2$ just dużo bardziej skomplikowana. Na razie ograniczymy się do zauważenia, że siatka równomierna jest fatalnym wyborem. Dokładniej, niech

$$\vec{t}_{j_1,\ldots,j_d} = [t^*_{j_1}, \ldots, t^*_{j_d}]^T, \qquad 1 \leqslant j_i \leqslant n, 1 \leqslant i \leqslant d,$$

będzie równomiernym rozkładem $N = n^d$ punktów w D. Rozpatrzenie prostokąta

$$[0, \tfrac{1}{2n}) \times \underbrace{[0,1) \times \cdots \times [0,1)}_{d-1}$$

wystarcza, aby się przekonać, że wtedy dyskrepancja wynosi co najmniej $\frac{1}{2n} = \frac{1}{2}N^{-1/d}$.

15.3. Błąd quasi-Monte Carlo

15.3.1. Formuła Zaremby

Jeśli $f : [0,1] \to \mathbb{R}$ jest funkcją różniczkowalną to dla każdego x mamy

$$f(x) = f(1) - \int_x^1 f'(t)\,dt = f(1) - \int_0^1 \mathbf{1}_{(x,1]}(t)\,f'(t)\,dt. \tag{15.2}$$

Wzór ten uogólnimy na przypadek funkcji wielu zmiennych w następujący sposób.

Najpierw wprowadzimy kilka niezbędnych oznaczeń. Dla podzbioru indeksów U,

$$\emptyset \neq U \subseteq \{1, 2, \ldots, d\},$$

niech $D_U = [0,1]^{|U|}$, gdzie $|U|$ jest liczbą elementów w U. Niech dalej $\vec{x}_U \in D_U$ będzie wektorem powstałym z wektora $\vec{x} = [x_1, x_2, \ldots, x_d]^T \in D$ poprzez usunięcie współrzędnych x_j z $j \notin U$, a $(\vec{x}_U; 1) \in D$ wektorem, którego j-ta współrzędna wynosi x_j dla $j \in U$ oraz wynosi 1 dla $j \notin U$. Na przykład, jeśli $d = 5$ i $U = \{1, 4\}$ to dla $\vec{x} = [x_1, x_2, x_3, x_4, x_5]^T$ mamy $\vec{x}_U = [x_1, x_4]^T$ i $(\vec{x}_U; 1) = [x_1, 1, 1, x_4, 1]^T$.

W końcu, niech

$$f'_U = \frac{\partial^{|U|} f}{\prod_{j \in U} \partial u_j}$$

będzie skrótowym zapisem odpowiedniej pochodnej mieszanej funkcji f.

Lemat 15.1. *Jeśli funkcja $f : D \to \mathbb{R}$ ma ciągłe pochodne cząstkowe mieszane f'_U dla wszystkich $\emptyset \neq U \subseteq \{1, 2, \ldots, d\}$ to*

$$f(\vec{x}) = f(\vec{1}) + \sum_U (-1)^{|U|} \int_{D_U} \mathbf{1}_{(\vec{x}_U, \vec{1}_U]}(\vec{z}_U)\, f'_U(\vec{z}_U; 1)\, d\vec{z}_U, \qquad \vec{x} \in D \tag{15.3}$$

($\vec{1} = [1, 1, \ldots, 1] \in \mathbb{R}^d$).

Dowód. Dowód przeprowadzimy przez indukcję względem d. Dla $d = 1$ równość (15.3) jest równoważna (15.2). Niech więc $d \geqslant 2$. Stosując założenie indukcyjne do f z ustaloną ostatnią współrzędną x_d mamy

$$f(\vec{x}) \;=\; f(\vec{x}_{\{d\}};1) \,+\, \sum_V (-1)^{|V|} \int_{D_V} \mathbf{1}_{(\vec{x}_V,\vec{1}_V]}(\vec{z}_V)\, f'_V(\vec{z}_V, x_d; 1)\, d\vec{z}_V,$$

gdzie sumowanie jest po wszystkich $\emptyset \neq V \subseteq \{1,2,\ldots,d-1\}$. Stosując dalej wzór (15.2) ze względu na x_d odpowiednio do $f(\vec{x}_{\{d\}};1)$ i $f'_V(\vec{z}_V, x_d; 1)$ dostajemy

$$\begin{aligned} f(\vec{x}) \;=\;& f(\vec{1}) - \int_{D_{\{d\}}} \mathbf{1}_{(\vec{x}_{\{d\}},\vec{1}_{\{d\}}]}\left(\vec{z}_{\{d\}}\right)\, f'_{\{d\}}\left(\vec{z}_{\{d\}};1\right)\, d\vec{z}_{\{d\}} \\ &+(-1)^{|V|} \int_{D_V} \mathbf{1}_{(\vec{x}_V,\vec{1}_V]}(\vec{z}_V)\, f'_V(\vec{z}_V;1)\, d\vec{z}_V \\ &+ \sum_{V\cup\{d\}} (-1)^{|V\cup\{d\}|} \int_{D_{V\cup\{d\}}} \mathbf{1}_{(\vec{x}_{V\cup\{d\}},\vec{1}_{V\cup\{d\}}]}\left(\vec{z}_{V\cup\{d\}}\right)\, f'_{V\cup\{d\}}\left(\vec{z}_{V\cup\{d\}};1\right)\, d\vec{z}_{V\cup\{d\}}. \end{aligned}$$

Teraz wystarczy porównać otrzymaną formułę do prawej strony (15.3). Drugi składnik odpowiada $U = \{d\}$, trzeci składnik podzbiorom $U \neq \emptyset$ do których nie należy d, a czwarty podzbiorom U takim, że $d \in U$ i $|U| \geqslant 2$. □

Zauważmy, że rozwijając $f(\vec{x})$ i $f(\vec{t}_j)$ zgodnie ze wzorem (15.3) mamy

$$\int_D f(\vec{x})\, d\vec{x} \;=\; f(\vec{1}) + \sum_U (-1)^{|U|} \int_{D_U} \left(\int_D \mathbf{1}_{(\vec{x}_U,\vec{1}_U]}(\vec{z}_U)\, d\vec{x} \right) f'_U(\vec{z}_U;1)\, d\vec{z}_U,$$

$$\frac{1}{N}\sum_{j=1}^N f(\vec{t}_j) \;=\; f(\vec{1}) + \sum_U (-1)^{|U|} \int_{D_U} \left(\frac{1}{N}\sum_{j=1}^N \mathbf{1}_{((\vec{t}_j)_U,\vec{1}_U]}(\vec{z}_U) \right) f'_U(\vec{z}_U;1)\, d\vec{z}_U.$$

Ponieważ wartość funkcji charakterystycznej odcinka $(\vec{a},\vec{b}]$ w punkcie $\vec{c}$ jest równa wartości funkcji charakterystycznej odcinka $[\vec{0},\vec{c})$ w punkcie $\vec{a}$ to

$$\int_D \mathbf{1}_{(\vec{x}_U,\vec{1}_U]}(\vec{z}_U)\, d\vec{x} \;=\; \int_{D_U} \mathbf{1}_{[\vec{0}_U,\vec{z}_U)}(\vec{x}_U)\, d\vec{x}_U \;=\; \prod_{j\in U} z_j,$$

$$\frac{1}{N}\sum_{j=1}^N \mathbf{1}_{((\vec{t}_j)_U,\vec{1}_U]}(\vec{z}_U) \;=\; \frac{1}{N}\,\#\left\{j \,:\, (\vec{t}_j)_U \in [\vec{0}_U,\vec{z}_U)\right\} \;=\; \frac{1}{N}\,\#\left\{j \,:\, \vec{t}_j \in \left[\vec{0},(\vec{z}_U;1)\right)\right\}.$$

Stąd otrzymujemy następującą *formułę Zaremby* na błąd quasi-Monte Carlo:

$$I_d(f) - \mathrm{QMC}_{d,N}(f) \;=\; \sum_U (-1)^{|U|} \int_{D_U} \mathrm{DISC}_d\left((\vec{z}_U;1);\vec{t}_1,\ldots,\vec{t}_N\right) \cdot f'_U(\vec{z}_U;1)\, d\vec{z}_U. \tag{15.4}$$

15.3.2. Nierówność Koksmy-Hlawki

Oznaczmy przez $\mathcal{V}_d(D)$ klasę funkcji $f : D \to \mathbb{R}$, których pochodne mieszane f'_U istnieją i są ciągłe dla wszystkich $\emptyset \neq U \subseteq \{1,\ldots,d\}$.

Definicja 15.3. *Wahaniem (w sensie Hardy-Krause)* funkcji $f \in \mathcal{V}_d(D)$ nazywamy wielkość

$$V_d(f) \;:=\; \sum_U \int_{D_U} \left| f'_U(\vec{z}_U;1) \right|\, d\vec{z}_U.$$

Zauważmy, że dla $d = 1$,

$$V_1(f) \;=\; \int_0^1 |f'(x)|\,dx \;=\; \sup\left\{ \sum_{j=1}^{k} |f(z_j) - f(z_{j-1})| \;:\; k \geqslant 1, 0 = z_0 < z_1 < \ldots < z_k = 1 \right\}$$

jest wahaniem funkcji w klasycznym sensie. Następujące bardzo ważne oszacowanie błędu metody quasi-Monte Carlo nosi nazwę *nierówności Koksmy-Hlawki.*

Twierdzenie 15.1. *Jeśli $f \in \mathcal{V}_d(D)$ to błąd metody quasi-Monte Carlo*

$$\mathrm{QMC}_{d,N}(f) = \frac{1}{N}\sum_{j=1}^{N} f(\vec{t}_j)$$

dla aproksymacji całki $I_d = \int_D f(\vec{x})\,d\vec{x}$ szacuje się przez

$$\left| I_d(f) - \mathrm{QMC}_{d,N}(f) \right| \;\leqslant\; \mathrm{DISC}_d^*(\vec{t}_1, \ldots, \vec{t}_N) \cdot V_d(f).$$

Dowód. Nierówność wynika natychmiast z formuły Zaremby (15.4), bowiem

$$\begin{aligned} |I_d(f) - \mathrm{QMC}_{d,N}(f)| \;&\leqslant\; \sum_U \int_{D_U} \left|\mathrm{DISC}_d\left((\vec{z}_U; 1); \vec{t}_1, \ldots, \vec{t}_N\right)\right| \cdot |f'_U(\vec{z}_U; 1)|\; d\vec{z}_U \\ &\leqslant\; \mathrm{DISC}_d^*(\vec{t}_1, \ldots, \vec{t}_N) \cdot \sum_U \int_{D_U} |f'_U(\vec{z}_U; 1)|\; d\vec{z}_U. \end{aligned}$$

□

W nierówności Koksmy-Hlawki czynnik błędu $V_d(f)$ zależny jedynie od funkcji jest oddzielony od czynnika błędu $\mathrm{DISC}_d^*(\vec{t}_1, \ldots, \vec{t}_N)$ zależnego od wyboru punktów. O ile nie mamy wpływu na wahanie funkcji, możemy starać się wybrać punkty $\vec{t}_j$ tak, aby zminimalizować ich dyskrepancję. Zasadnicze pytanie brzmi: jak mała może być dyskrepacja? W szczególności, czy można wybrać nieskończony ciąg punktów tak, że oparte na nim algorytmy quasi-Monte Carlo pokonują przekleństwo wymiaru w klasie $\mathcal{V}_d(D)$?

Na miejscu jest teraz uwaga, że dzięki obecności pochodnych mieszanych rozumowanie analogiczne do tego z dowodu twierdzenia 13.4 prowadzi w klasie funkcji $f \in \mathcal{V}_d(D)$ z $V(f) \leqslant 1$ jedynie do oszacowania z dołu błędu przez cN^{-1} (a nie $cN^{-r/d}$).

Okazuje się, że pełne odpowiedzi na zadane pytania nie są znane. Najlepsze ciągi nieskończone $\vec{t}_1, \vec{t}_2, \ldots, \vec{t}_n, \ldots$ spełniają nierówność

$$\mathrm{DISC}_d^*\left(\vec{t}_1, \ldots, \vec{t}_N\right) \;\leqslant\; C_d \frac{\ln^d N}{N}, \qquad N = 1, 2, 3, \ldots,$$

gdzie $C_d > 0$ nie zależy od N.

Wydaje się więc, że w klasie $\mathcal{V}_d(D)$ metody quasi-Monte Carlo dają istotnie lepsze wyniki od Monte Carlo, bo błąd nie tylko jest deterministyczny, ale też dla $f \in \mathcal{V}_d(D)$ zbiega do zera dużo szybciej, tzn. błąd jest rzędu $N^{-1+\epsilon}$ dla dowolnego $\epsilon > 0$ w przypadku quasi-Monte Carlo, oraz $N^{-1/2}$ w przypadku Monte Carlo. Nie do końca jest to prawdą. Zauważmy bowiem, że w praktycznych obliczeniach wnioski asymptotyczne nie mają zastosowania, gdy wymiar d jest bardzo duży. Wtedy czynnik $C_d \ln^d N$ może mieć istotne znaczenie i wręcz sprawiać,

że nierówność Koksmy-Hlawki staje się bezużyteczna. Dodatkowo, dobre oszacowanie C_d jest wyjątkowe trudne.

A jednak metody quasi-Monte Carlo są z powodzeniem stosowane w praktyce obliczeniowej nawet dla dużych wymiarów d. Istnieje wiele hipotez tworzonych w celu wyjaśnienia tego pozornego paradoksu. Jedna z najbardziej popularnych i już dość dobrze uzasadnionych teoretycznie mówi, że w praktyce mamy co prawda do czynienia z funkcjami bardzo wielu zmiennych, ale istotnych jest jedynie kilka zmiennych albo grupy kilku zmiennych. Matematycznie oznacza to, że w odpowiadających tym założeniom przestrzeniach zachodzą dużo mocniejsze odpowiedniki nierówności Koksmy-Hlawki.

15.4. Ciągi o niskiej dyskrepancji

Definicja 15.4. Ciąg nieskończony $\vec{t}_1, \vec{t}_2, \vec{t}_3, \ldots$ nazywamy *ciągem o niskiej dyskrepancji* jeśli istnieje $C_d > 0$ taka, że dla wszystkich N

$$\mathrm{DISC}_d^*\left(\vec{t}_1, \ldots, \vec{t}_N\right) \leqslant C_d \frac{\ln^d N}{N}.$$

Istnieje bardzo dużo efektywnych konstrukcji ciągów punktów o niskiej dyskrepancji. Teraz poznamy jedynie kilka najbardziej popularnych z nich.

15.4.1. Ciąg Van der Corputa

Niech $b \geqslant 2$ będzie ustaloną liczbą naturalną. Wtedy dowolną liczbę naturalną n można jednoznacznie przedstawić w postaci

$$n = \sum_{j=0}^{\infty} a_j(n)\, b^j,$$

gdzie $a_j \in \{0, 1, \ldots, b-1\}$ i tylko dla skończonej liczby indeksów j mamy $a_j \neq 0$. Mówiąc inaczej, a_j są kolejnymi cyframi rozwinięcia liczby n w bazie b. Wykorzystując powyższe rozwinięcie funkcję *radykalnej odwrotności* $\psi_b : \{0, 1, 2, \ldots\} \to [0, 1)$ definujemy jako

$$\psi_b(n) := \sum_{j=1}^{\infty} a_j(n)\, b^{-(j+1)}.$$

Na przykład, dla bazy $b = 2$ mamy $13 = 2^0 + 2^2 + 2^3 = (1101)_2$, czyli $\psi_2(13) = \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = \frac{11}{16}$.

Kolejne wartości radykalnej odwrotności,

$$\psi_b(1), \psi_b(2), \ldots, \psi_b(n), \ldots,$$

tworzą jednowymiarowy ciąg *Van der Corputa*. Dla $b = 2$ kolejne punkty ciągu wynoszą więc

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}, \frac{13}{16}, \frac{3}{16}, \frac{11}{16}, \frac{7}{16}, \frac{15}{16}, \frac{1}{32}, \frac{17}{32}, \ldots$$

Ciąg Van der Corputa jest ciągiem o niskiej dyskrepancji dla dowolnie dobranej bazy b, tzn. dyskrepancja N początkowych wyrazów szacuje się z góry przez $C_1 \ln N/N$. Zauważmy jednak, że dla $N = b^k - 1$, $k \geqslant 1$, dostajemy równomierne rozmieszczenie punktów, tzn. tworzą one zbiór $\{j/(N+1) : 1 \leqslant j \leqslant N\}$, którego dyskrepancja wynosi $(N+1)^{-1}$. Stąd, pożądane są raczej mniejsze bazy b; im większe b tym rzadziej ze względu na N osiągana jest dyskrepancja proporcjonalna do $1/N$.

15.4.2. Konstrukcje Haltona i Sobol'a

Jednowymiarowy ciąg Van der Corputa jest podstawą konstrukcji wielu ciągów w wyższych wymiarach d. Jedna z nich prowadzi do ciągu *Haltona* $\{\vec{h}_k\}_{k\geqslant 1}$, którego k-ty punkt wynosi

$$\vec{h}_k \;=\; [\psi_{b_1}(k), \psi_{b_2}(k), \ldots, \psi_{b_d}(k)]^T\,.$$

Liczby $b_1, \ldots, b_d$ są tu danymi bazami. Od razu zauważamy, że wybór $b_1 = \cdots = b_d$ nie jest dobry, bo prowadzi do siatki równomiernej w $[0,1)^d$, zob. rozdział 15.2. Jeśli jednak b_i są liczbami pierwszymi to $\{\vec{h}_k\}_{k\geqslant 1}$ jest już ciągiem o niskiej dyskrepancji.

Minusem konstrukcji Haltona jest to, że dla dużych wymiarów d bazy b_d też są duże co, jak wcześniej zauważyliśmy, nie jest korzystne. Problemu tego unika bardziej skomplikowana konstrukcja *Sobol'a*, gdzie na każdej zmiennej pracujemy z tą samą bazą $b = 2$.

Ideę konstrukcji ciągu Sobol'a $\{\vec{s}_k\}_{k\geqslant 1}$ przedstawimy najpierw zakładając $d = 1$. Niech

$$\vec{a}(k) = [a_0(k), \ldots, a_{r-1}(k)]^T$$

będzie wektorem kolejnych bitów w rozwinięciu dwójkowym liczby k,

$$k \;=\; a_0(k) + 2a_1(k) + \cdots + 2^{r-1}a_{r-1}(k).$$

Wtedy

$$s_k \;=\; \frac{y_1(k)}{2} + \frac{y_2(k)}{4} + \cdots + \frac{y_r(k)}{2^r},$$

gdzie

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_r(k) \end{bmatrix} \;=\; V \cdot \begin{bmatrix} a_0(k) \\ a_1(k) \\ \vdots \\ a_{r-1}(k) \end{bmatrix} \mod 2,$$

a V jest specjalnie dobraną macierzą o elementach 0 i 1, zwaną macierzą generującą. (Zauważmy, że jeśli V jest identycznością to otrzymany ciąg jest ciągiem Van der Corputa.)

Oznaczając $V = [\vec{v}_1, \ldots, \vec{v}_r]$ możemy równoważnie zapisać

$$\vec{y}(k) \;=\; a_0(k)\cdot\vec{v}_1 \oplus \cdots \oplus a_{r-1}(k)\cdot\vec{v}_r,$$

gdzie $\oplus$ jest operacją XOR, czyli dodawaniem bitów modulo 2,

$$0\oplus 0 = 0, \quad 0\oplus 1 = 1, \quad 1\oplus 0 = 1, \quad 1\oplus 1 = 0.$$

Dla $d \geqslant 2$, kolejne współrzędne punktu $\vec{s}_k$ wyliczane są według powyższej recepty, ale używając różnych macierzy generujących V. I właśnie problem wyboru macierzy generujących tak, aby otrzymać ciąg o niskiej dyskrepancji jest istotą konstrukcji Sobol'a. Dodajmy, że jest to problem wysoce nietrywialny.

15.4.3. Sieci (t, m, d) i ciągi (t, d)

Dla $b \geqslant 2$ definiujemy b-prostokąt w $[0,1)^d$ jako

$$\left[\frac{a_1}{b^{j_1}}, \frac{a_1-1}{b^{j_1}}\right) \times \cdots \times \left[\frac{a_d}{b^{j_d}}, \frac{a_d-1}{b^{j_d}}\right),$$

gdzie $j_i \in \{0,1,2,\ldots\}$, $a_i \in \{0,1,\ldots,b^{j_i-1}\}$, $1 \leqslant i \leqslant d$. Na przykład, jeśli $d = 1$ i $b = 2$ to przedziały $[3/4, 1)$ i $[3/4, 7/8)$ są b-prostokątami, ale nie jest nim $[5/8, 7/8)$. Zauważmy, że objętość b-prostokąta wynosi $b^{-(j_1+\cdots+j_d)}$.

Definicja 15.5. Niech $b \geqslant 2$ i $0 \leqslant t \leqslant m$. Siecią (t, m, d) w bazie b nazywamy zbiór b^m punktów w $[0,1)^d$ o własności, że w każdym b-prostokącie o objętości b^{t-m} znajduje się dokładnie b^t punktów tego zbioru.

Mówiąc inaczej, sieć (t, m, d) dokładnie pokazuje objętości b-prostokątów poprzez iloraz b^t/b^m liczby punktów, które należą do prostokąta do liczby wszystkich punktów.

Definicja 15.6. Ciąg nieskończony $\vec{t}_1, \vec{t}_2, \ldots$ w $[0,1)^d$ nazywamy ciągiem (t, d) w bazie b jeśli dla wszystkich $m > t$ i $j = 0, 1, 2, \ldots$ segment

$$\left\{ \vec{t}_i \,:\, jb^m < i \leqslant (j+1)b^m \right\}$$

jest siecią (t, m, d) w bazie b.

Następujące twierdzenie jest podstawą wielu konstrukcji ciągów o niskiej dyskrepancji.

Twierdzenie 15.2. *Każdy ciąg (t, d) w bazie b jest ciągiem o niskiej dyskrepancji.*

Pokazanie konkretnych konstrukcji ciągów (t, d) wykracza poza ramy tego wykładu. Powiemy tylko, że szczególne wybory macierzy generujących w konstrukcji Sobol'a prowadzą do ciągów (t, d). Inne konstrukcje należą do Faure'a, Niederreitera, Tezuki i in.

Literatura

[1] Steven F. Ashby, Thomas A. Manteuffel, Paul E. Saylor. A taxonomy for conjugate gradient methods. *SIAM J. Numer. Anal.*, 27(6):1542–1568, 1990.

[2] James W. Demmel. *Applied numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[3] John E. Dennis Jr., Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice-Hall Series in Computational Mathematics. Prentice-Hall Inc., Englewood Cliffs, N.J., 1983.

[4] Peter Deuflhard. *Newton methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms.* Springer International, 2002.

[5] Peter Deuflhard, Andreas Hohmann. *Numerical analysis in modern scientific computing*, wolumen 43 serii *Texts in Applied Mathematics.* Springer-Verlag, New York, wydanie ii, 2003. An introduction.

[6] Eugene G. Dyakonov. *Optimization in solving elliptic problems.* CRC Press, Boca Raton, FL, 1996. Translated from the 1989 Russian original, Translation edited and with a preface by Steve McCormick.

[7] Gene H. Golub, Charles F. Van Loan. *Matrix computations.* Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, wydanie iii, 1996.

[8] J.M. Jankowscy, M. Dryja. *Przegląd metod i algorytmów numerycznych, tom I i II.* Biblioteka inżynierii oprogramowania. Wydawnictwo Naukowo-Techniczne, Warszawa, 1995.

[9] C. T. Kelley. *Iterative methods for linear and nonlinear equations*, wolumen 16 serii *Frontiers in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. With separately available software.

[10] A. Kiełbasiński, H. Schwetlick. *Numeryczna algebra liniowa.* Wydawnictwa Naukowo-Techniczne, 1992.

[11] David Kincaid, Ward Cheney. *Analiza numeryczna.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2006. Tłum. z ang.: S. Paszkowski.

[12] J. M. Ortega, W. C. Rheinboldt. *Iterative solutions of nonlinear equations in several variables.* Academic Press, New York, 1970.

[13] Yousef Saad. *Iterative methods for sparse linear systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, wydanie ii, 2003.

[14] Barry F. Smith, Petter E. Bjørstad, William D. Gropp. *Domain decomposition.* Cambridge University Press, Cambridge, 1996. Parallel multilevel methods for elliptic partial differential equations.

[15] J. Stoer, R. Bulirsch. *Wstęp do analizy numerycznej.* Biblioteka matematyczna. PWN, Warszawa, 1995.

[16] Andrea Toselli, Olof Widlund. *Domain decomposition methods—algorithms and theory*, wolumen 34 serii *Springer Series in Computational Mathematics.* Springer-Verlag, Berlin, 2005.

[17] J. F. Traub. *Iterative Methods for the Solution of Equations.* Englewood Cliffs, New York, 1964.